PandoGen: Generating complete instances of future SARS-CoV2 sequences using Deep Learning

Anand Ramachandran[†], Steven S. Lumetta, Deming Chen
University of Illinois at Urbana-Champaign

Abstract

Deep generative models have achieved breakthrough performance in generating computer code, instances of human language and images. We explore the use of these models to create as yet undiscovered instances of viral sequences in a pandemic situation. Towards this goal, we formulate a novel framework for training models to align the sequence generation problem to the characteristics of a pandemic. We applied our method to modeling the SARS-CoV2 Spike protein, the primary driver of the COVID-19 pandemic, and compared our method to models trained via prevalent practices applied to biological sequence modeling. Our method substantially outperforms a state-of-the-art generative model finetuned on SARS-CoV2 data, producing samples containing sequences which are four times as likely to be real, undiscovered sequences, and ten times as infectious. Our method can forecast novel lineages that will be reported over 10 weeks in the future. Given a limited sequence budget, our method also generates sequences belonging to the Delta variant and multiple dominant Omicron subvariants up to a month in advance.

1 Introduction

Deep generative models have achieved breakthrough results in generating complete, self-contained, and realistic examples of complex data such as images [RPG+21], language [Ope23], and computer code [LCC+22]. A class of models called Large Language Models (LLMs) have recently achieved groundbreaking performances in following human instructions faithfully [Ope23] [OWJ+22] and performing at the human-level in competitive programming [LCC+22]. Models based on LLM architectures have recently gained popularity in modeling biological sequences as well [RMS+19] [EHD+21] and can be useful tools in generating samples with interesting properties [FSH22] [SRK+21]. When applied to proteins these models are referred to as Protein Language Models (PLMs) in literature.

In this article, we address a hitherto unaddressed question: how can a PLM be trained to generate complete, self-contained, realistic, and as-yet-undiscovered instances of biological sequences in a pandemic situation, which may appear in the future? We try to answer this question for the SARS-CoV2 virus which caused the COVID-19 pandemic. Among SARS-CoV2 proteins, the Spike protein holds special significance. The Spike protein binds to the ACE-2 enzyme found on the cell surface of the host, gaining entrance to the cell. The efficiency of this process reflects the infectiousness of the virus [Xia21]. As a result, many treatment methods such as antibodies target the Spike protein for neutralization. Advance knowledge of future Spike protein sequences can hence provide an important resource for stress-testing interventions such as vaccines and antibody treatments. Hence we focus our efforts on the Spike protein.

Deep Mutational Scanning (DMS) [FF14] has become popular as a method to probe properties of protein sequences, and offers one avenue for performing tests of hypothetical Spike protein sequences. Recently, a DMS method was developed that can probe the entirety of the Spike protein sequence [DCR+23], capable of scanning sequence sets numbering to the order of 10^5. Through experiments such as these, a Spike protein variant's ability to infiltrate the cell through receptor binding, as well as its ability to escape or withstand antibody binding and neutralization can be studied. However, simply the number of combinations of all known mutations of the Spike protein is astronomical. Given limited sequence budgets, we need to obtain a set of good quality candidates to test through methods such as DMS. Deep generative models, in principle, can bridge this gap by providing plausible sequences to test through DMS.

[†]Work does not relate to author's position at Amazon

However, in practice, it is not clear how the models can be trained to achieve this. Deep generative models are typically used to parameterize the probability distribution of a given dataset, such as sequences. Traditionally, models are trained using a known set of sequences related to a task. The models learn the distribution of the training data, by maximizing the likelihood of the training examples. In a pandemic situation, we are interested in as yet undiscovered sequences which will potentially be found in the future. There are distinctions between sequences known until a point in a pandemic, and new sequences to come. For example, as the pandemic progresses, sequences accumulate more mutations. So, future sequences likely carry a larger number of mutations than the sequences on which models are trained. Sampling from such models, which are trained on known sequences, hence, is not a good way to model such *arrow of time* effects.

Also, model predictions must be *salient*. In the context of SARS-CoV2, this means that sampled sequences must have non-trivial case counts. For instance, out of all Spike protein sequences submitted to GISAID [SM17] as of 2022-11-29, there are 762,192 unique sequences. Out of these, only 47,839 sequences (or 6.2%) had been reported 10 or more times in the database and 523,582 (or 69%) had been reported exactly once. Given that a super-majority of sequences were only reported once, the models need to know that some sequences are more important than others, but at the same time, that all sequences carry important information about plausible structure and function related to the organism. A simple way to include this information in training is to have the training set reflect the true distribution of the underlying data, by repeating sequences in the training set in proportion to their GISAID case counts. This would teach the model which sequences are more salient, and what are their characteristics. However, this does not answer the question of how the distributional characteristics found within existing sequences can be extrapolated to produce future sequences.

Recently, a method called Reinforcement Learning Through Human Feedback (RLHF) has become popular for training LLMs. Here, a discriminative model is trained to learn how humans would score different generative model responses to human queries [OWJ+22] [Ope23]. For this, human-labeled data is used to train the discriminative model. Then, the generative model is trained to get better "rewards" from the discriminative model. While this approach produces superior results in Natural Language tasks in many contexts, it does not fit our requirements directly. First, RLHF is used for improving a model's response to a human-provided prompt or query. That is, the human query acts as a prefix for which the model generates a continuation. In our problem, there is no such prompt, as the model is expected to produce the full protein sequence. Second, it is not clear what replaces human annotations for protein sequences. Finally, it is not clear how the arrow of time effects we discussed above can be addressed wherein the generative model needs to produce sequences different from the training distribution, while retaining important characteristics.

We formulate a novel solution with two parts to address these issues. First, we train a discriminative model that predicts the infectiousness potential of a Spike protein from its sequence. It does this purely using information available in SARS-CoV2 repositories, avoiding expensive actions such as labeling or laboratory experiments. Viewing the spread of the disease as games played among the different Spike sequences, the discriminative model learns the probability distribution of the outcome in pairwise competitions among Spike sequences. The competitions are setup so that sequences with higher case counts in GISAID have a higher chance to win. However, a sequence discovered under one set of circumstances (e.g., available sequencing capacity, disease penetration, treatments and other preventative measures) cannot be fairly compared with a sequence discovered under a different set of circumstances. To avoid bad comparisons, during training only "fair" games are played. Second, we teach a generative model, patterns related to successful sequences, and to apply this knowledge to the generation of new sequences, rather than sequences from the training distribution. To achieve this, we use the discriminative model to guide the finetuning of the generative model where (1) the generative model learns to generate successful sequences and (2) the generative model preserves sequence understanding it learnt from the training set, but learns to avoid generating sequences from the training set. Together, these steps help us to align the problem of sequence generation to the requirements of a pandemic scenario.

We use a type of generative models called deep autoregressive models, which are among the most successful sequence generative models today [RNS+18] [RWC+19] [ODZ+16]. Deep autoregressive models implement sequence generation through iterative generation of the next sequence element. Hence the likelihood of a complete sequence may be written as follows.

$$P(X_{1:L}) = \Pi_{i=1}^{L} P(x_i \mid X_{1:i-1}; \Phi) \tag{1}$$

Here, the probability of a full sequence, $X_{1:L}$, is written as the product of the probability of each sequence element conditioned on the past sequence elements, $P(x_i \mid X_{1:i-1}; \Phi)$. Φ represents the model's parameters. The underlying neural network can have hundreds of millions or billions of parameters, resulting in highly capable modeling. When training large neural networks, it is standard practice to pre-train them on unlabeled data in the domain. Such data is cheap to obtain due to lack of expensive processing steps such as labeling. For example, in NLP, data from the internet [RSR+20] [DSM+21] is used to pre-train large models. This pre-training step allows models to gain an understanding of the structure of the data in the domain of interest. Pre-training seeks to maximize the likelihood of the training data under the distribution parameterized by the model. That is, the goal is to estimate model parameters, Φ^* which satisfy the following [DEKM98].

$$\Phi^* = \arg\max_{\Phi} \log P(\mathcal{D} \mid \Phi) = \arg\max_{\Phi} \sum_{X^{(j)} \in \mathcal{D}} \sum_{i=1}^{L(j)} \log P(x_i^{(j)} \mid X_{1:i-1}^{(j)}; \Phi) \tag{2}$$

Here, $L(j)$ is the length of the sequence X_j, i iterates over the sequence elements in sequence X_j, and j iterates over the full dataset. Hence this equation indicates the search for a model parameter set, Φ^* under which the log-likelihood of the dataset, $\mathcal{D}$, is maximized. In practice, for deep neural networks, the training is implemented through mini-batch gradient descent.

We formulate a four-stage training pipeline for our pandemic sequence generator based on deep autoregressive models. Following pretraining approaches normally adopted for such models, the first stage in our pipeline is to train a deep autoregressive model on the UniProt dataset [uni17], specifically the UniRef50 sequence set, which contains tens of millions of general protein sequences. We used a version of the database prior to the onset of the COVID-19 pandemic.

The second stage is to finetune the model on SARS-CoV2 Spike protein sequences from GISAID, producing a SARS-CoV2 deep autoregressive (SDA) model. We select a date cut-off point and only sequences reported on or before this date were used for this finetuning step. This step also proceeds via likelihood maximization. This method, called Transfer Learning [RSR+20], is the canonical approach to finetune large models for a sub-domain (here, SARS-CoV2) by transferring knowledge from a large model trained on domain-level data (UniProt sequences).

There exist similar works in literature for stages one and two. For instance, Prot GPT2 [FSH22] is a large protein generative model trained on publicly available protein corpuses using likelihood maximization techniques. The model is derived from the popular GPT2 Large Language Model (LLM) that is popular in NLP [RWC+19], and is a deep autoregressive model. The method has also been fine-tuned on Spike protein sequences similar to the second stage outlined above via Transfer Learning [Dho23]. In the latter case, the model was asked to generate continuations of a specific localized sequence context within the Spike protein. The generated results were analyzed for individual mutations within the context of interest. However, the work does not examine the question of *de novo* generation of complete Spike protein sequences, even though, Prot GPT2 is capable of generating complete instances of the Spike protein. In this article, we compare our results to that from a finetuned Prot GPT2 operated as a *de novo* sequence generator.

The third stage in our training pipeline is to train a reward model to predict the potential success or infectiousness of a Spike sequence, as outlined above, by setting up "fair", pairwise competitions among SARS-CoV2 sequences reported in the training period, and using the reward model to predict the winner. This stage uses the same training sequences as the SDA model, along with the geographical location, and the date of reporting. Training uses only information from the training period. So if a sequence was discovered in the training period, only the instances of the sequence reported within the training period are used; if the sequence continues to be reported in the future, that information is not used.

The fourth stage (Figure 1) is to finetune the SDA model directly in its operational setting (i.e., sequence generation), guided by the reward model. An iterative procedure is used here, where the SDA model generates sequences, each generated sequence is marked as known (based on what is known in the training period) or unknown, the reward model is run on each generated sequence to mark its infectiousness potential, and this annotated data is used to finetune the SDA model. The goal is to

improve the SDA model's ability to generate novel sequences with high rewards. Note that no new external training data is used in the fourth-stage.

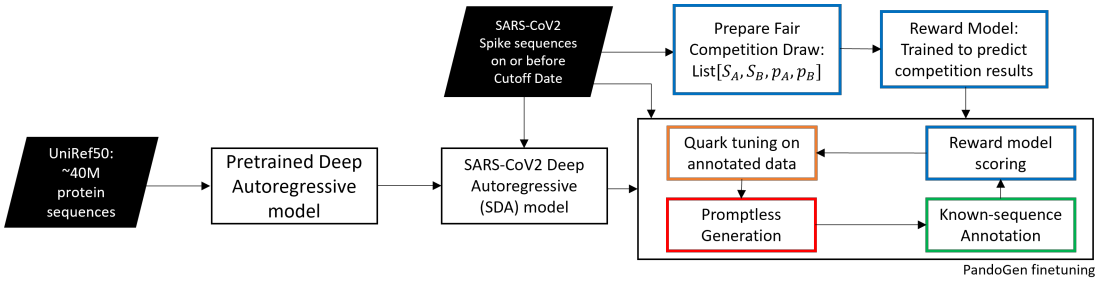

Figure 1: The PandoGen high-level training flow. The training flow starts with pre-training on the UniProt database, followed by finetuning on the SARS-CoV2 sequences to produce an SDA model. We then create a competition draw among known sequences based on the fair-competition principle mentioned in Section 1 (detailed in Section 4.5) and train a reward model to learn the probability distribution of the outcome for games between two sequences. The SDA model is then finetuned in its operational setting through an iterative procedure involving promptless generation, historical data look-up, reward model scoring, and tuning using the generated annotated data. To perform model updates during training, we use a method based on Quark [LWJ+22]. The iterative process is terminated when reward improvements saturate.

As a result of this finetuning, we find that the resultant PandoGen model produces sequence samples with higher quality. The samples have higher fraction of real sequences among novel sequences, higher number of novel sequences per sample, and higher cumulative case counts than sequences from a finetuned Prot GPT2 model. For several weeks after the training period, the model is able to predict a significant fraction of viral lineages before they are reported in GISAID; novel lineages are predicted up to 10 weeks in advance. We also trained PandoGen models using data available many days before the Delta variant, and multiple Omicron subvariants were first reported in GISAID. When generating sequences from these trained models within a sequence budget, we find that PandoGen consistently forecasts many of these variants ahead of time.

2 Results

We present two sets of results: 1) Quantitative comparisons of PandoGen with baseline methods and 2) Testing the efficacy of PandoGen in forecasting important COVID-19 variants.

2.1 Quantitative comparisons

To examine efficacy of PandoGen in generating future sequences quantitatively, we setup an experiment where models including PandoGen were trained on sequences reported until a designated cutoff date (the "training period"). The methods are then used to generate sequences, and the efficacy of the generated samples is determined based on sequences reported after the cutoff date. As multiple cases may be reported for a given Spike protein sequence in GISAID in the training period, we distinguish between a training set which contains each sequence only once ("unique sequences") and a training set which contains each sequence as many times as it is reported in the training period in GISAID ("all sequences"). We train four models (1) the SDA model trained on unique sequences in the training period (2) Prot GPT2 finetuned on unique sequences in the training period ("Prot GPT2 unenumerated" model) (3) Prot GPT2 finetuned on all sequences in the training period ("Prot GPT2 enumerated" model) and (4) PandoGen. Sequences were sampled from each model using multiple sampling configurations. Each model and sampling configuration was run six independent times, each time generating 2048 sequences. For this experiment, the training period is designated as on or before 2021-06-15.

2.1.1 Sampling algorithms and operating points

The unrestricted sampling process from a deep autoregressive model proceeds as follows

$$\hat{x}_i \sim P(x_i \mid X_{1:i-1}; \Phi) \tag{3}$$

Here, Φ represents the model parameters. As shown, the ith sequence element, x_i, is sampled from the model distribution, which is conditioned on all previous sequence elements sampled from the model. This is iteratively performed until the end of sequence is encountered. This sampling process yields sequences with the maximum amount of output variation from the trained model. On the other extreme, the following greedy search algorithm is also available, which always yields exactly one sequence, representing a minimal amount of variation in the model output.

$$\hat{x}_i = \arg\max_{x_i} P(x_i \mid X_{1:i-1}; \Phi) \tag{4}$$

There are different methods to tune the model's output so that it lies somewhere between the two extremes. We adopt nucleus sampling [HBD+19], a widely popular sampling method used for LLMs.

To explain nucleus sampling, we define a few terms for clarity first. A sequence is an ordered tuple of sequence elements. Each sequence element is a token that is sampled from the set of all possible tokens that can appear in the sequence, called the sequence alphabet. Nucleus sampling filters away noisy cases from the sampling process in Equation 3 by defining a nucleus, which is a subset of the sequence alphabet, and then sampling from within that nucleus at each step. The nucleus at step i is defined as follows.

$$N_i = \arg\min_{||M||} \Big(\sum_{x \in M} P(x \mid X_{1:i-1}; \Phi) \geq p \Big) \tag{5}$$

Here, M is a subset of the sequence alphabet. The nucleus, N_i, is the smallest subset of this set for which the sum of the probabilities at timestep i in the sequence generation process is at least p, a parameter of nucleus sampling. If multiple subsets of the same cardinality satisfy the condition, the subset, M^*, with the maximum sum $\sum_{x \in M^*} P(x \mid X_{1:i-1}; \Phi)$ can be selected to break the tie. The tokens in the sequence alphabet outside N_i are discarded, and the ith sequence element is sampled from within N_i (the probabilities within N_i are renormalized for this purpose). The maximum value of p is 1 where the nucleus is the complete sequence alphabet. As p decreases, fewer elements are kept inside the nucleus, reducing the number of potential outcomes for each sampling step in the sequence generation process. Essentially, this removes lower likelihood, potentially erroneous, elements from consideration at each sampling step. However, when p is sufficiently small, the nucleus will begin to exclude non-erroneous, high-information cases. So, at sufficiently low values of p, the model would fail to output new information. Hence, varying p allows us to explore the trade-off between error control, and novelty of information in the sampled output.

While p gives us a knob to tune the sampling operation, the value of p alone does not convey a meaningful characterization of the operating point of the model in terms of characteristics of its sequence sample. To provide this characterization, we look to quantify the difference between a generated sample and the known sequences of SARS-CoV2. A higher degree of difference indicates that generated sequences are likely further apart, evolutionarily, from known sequences. It is natural to expect the models to have a higher error rate when the sample difference with respect to the training set is higher, as models would need to predict further into the future in this case. We expect that lower values of p cause the sample difference to be lower and higher values allow generated sequences to deviate further from the training set.

To quantify sample difference from the known SARS-CoV2 corpus without expensive computational overheads, we formulate a method using k-mer matching. This is based on methods used to determine data quality in genome sequencing [HMRC21]. First, we collect all unique SARS-CoV2 spike sequences in the training period. We obtain all the k-mers in these sequences into a reference k-mer set ($k = 11$ in our experiments). Next, for each unique sequence in a generated sample, we count the number of k-mers in the sequence that are not found in the reference set. Using this, we find the average per-sequence k-mer novelty of the sample with respect to the reference k-mer set. We designate this quantity as the measure of sample difference with respect to the reference (training) sample. Specifically, if S_i is the number of k-mers in the ith sequence that are not found in the reference k-mer set, the k-mer difference

for the sample is $\frac{\sum_{i=1}^{N} S_i}{N}$, where N is the number of unique sequences in the sample. Naturally, the higher the k-mer difference of a sample, the further it is from the known set of SARS-CoV2 sequences.

2.1.2 Model comparisons

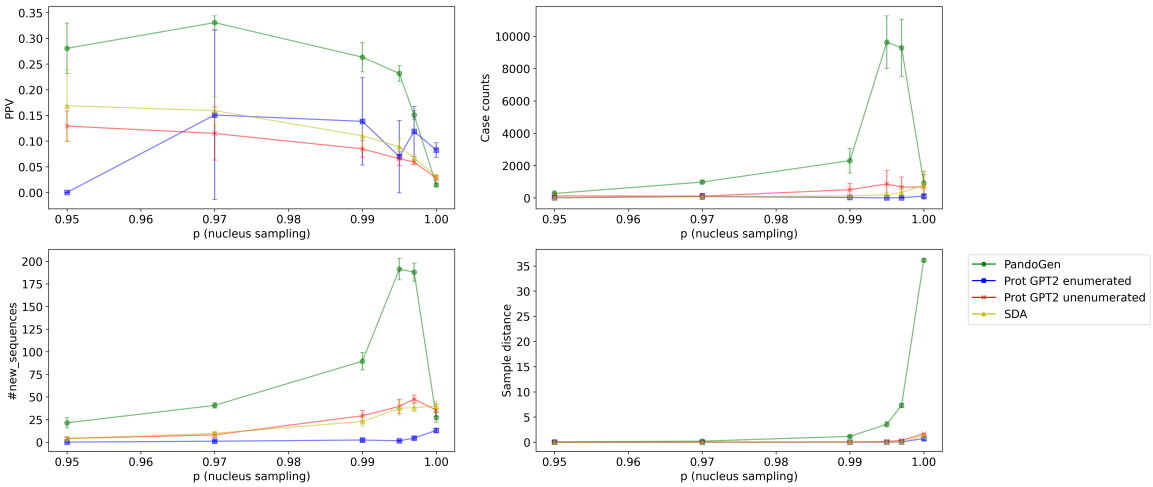

Figure 2: Sequence sample characteristics from the competing methods. PPV or Positive Predictive Value is calculated as the number of novel real sequences divided by the number of novel sequences in the sample, where novel sequences are sequences not reported in the training period. Case counts indicates the number of times novel sequences in the sample are reported in the GISAID database. "#new sequences" refers to the number of novel sequences in the sample that are found in GISAID. Sample distance indicates the k-mer difference between the sample and the training set. The spreads represent the 95% confidence intervals for the runs.

Figure 2 compares the characteristics of samples generated from the models for a range of values of p. We selected values of p between 0.95 and 1. The lower bound was chosen to be 0.95 because, at 0.95 the models output very few novel sequences (e.g., an average of 31 novel sequences out of 2048 generated sequences for the "Prot GPT2 unenumerated" case) and values lower than 0.95 are expected to produce even less novelty in the output, hence being uninteresting operating points. We look at the Positive Predictive Value (PPV) of novel sequences from the models, which is the fraction of real sequences out of generated non-training sequences, the case counts (number times a sequence is reported in GISAID) of non-training sequences, the number of new, real, sequences produced by the models, and the sample difference as detailed above. At $p = 0.95$, we see that most methods produce very few novel sequences as mentioned above. As p increases, the number of new sequences which are also real sequences produced by the models starts to increase, and the number produced by PandoGen shows the highest increase. PandoGen produces almost $4\times$ the number of new sequences compared to the second-best case. PandoGen asserts a bigger lead, close to $10\times$, for case counts. Overall, PandoGen produces samples with higher PPV, higher case counts, and higher number of novel sequences than other methods. PandoGen's performance at $p = 1$ shows a drop-off. This is because, at $p = 1$, PandoGen samples exhibit a stark increase in the sample difference. This means the model is trying to predict sequences very different from training sequences, which may be expected to occur further in the future. Making such predictions can be difficult and unreliable. We recommend using sample k-mer difference to filter out samples which are too ambitious and hence unreliable or error-prone.

Below, we examine sequence generation metrics as a function of sequence rank. As generative sequence models, all methods assign probabilities to generated samples. Sequence ranking is determined based on this probability value. Higher ranking sequences are expected to have better quality, and focusing on higher ranking sequences is a way to get the most reliable data out of a sample set for downstream analyses.

Figure 3 shows the fraction of real sequences among the top-1000 novel sequences generated from the models (or the PPV of novel sequences), as a function of sequence rank. A sequence is real, if it was

reported in the GISAID database at any point in time. If a sequence is real but first reported in the training period, it is not considered for the plot. Higher values of PPV indicate higher trustworthiness of the model. As may be seen, multiple PandoGen configurations far outperform the other methods with more than twice the PPV for the highest ranked sequences.

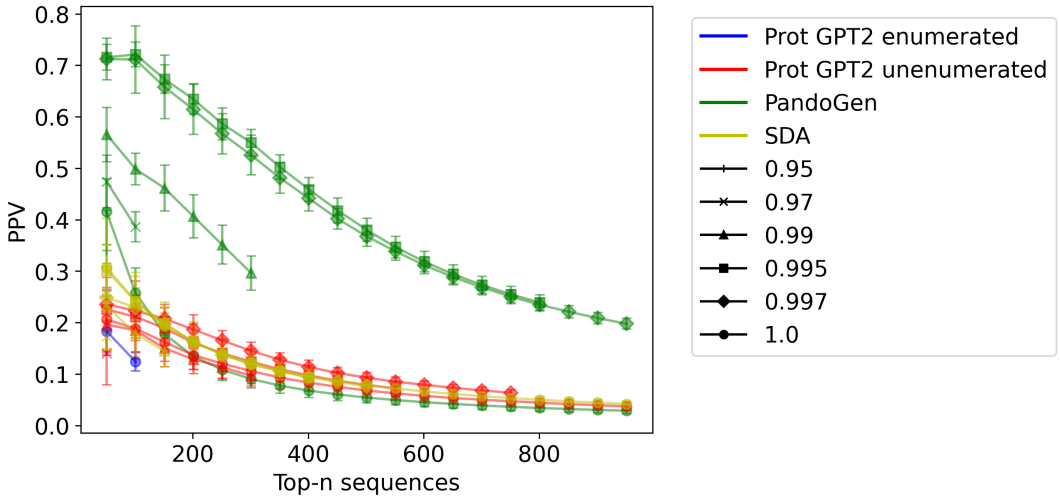

Figure 3: PPV by sequence rank for the top 1000 new sequences produced by the different methods. Error bars represent the 95% confidence intervals for six independent sampling runs.

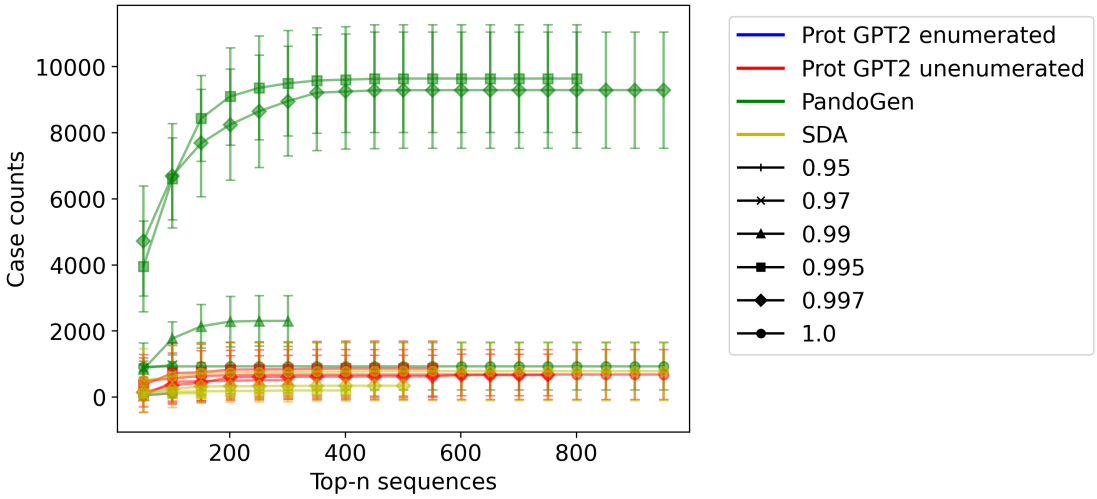

Figure 4: GISAID case count by sequence rank for the top 1000 new sequences produced by the different methods. Error bars represent the 95% confidence intervals for six independent sampling runs.

Figure 4 shows the cumulative number of GISAID cases of the generated novel sequences from the models (sequences not reported in the training period), as a function of sequence rank. Multiple PandoGen configurations dominate the plot, with close to 10 times the case count of competing methods for top-1000 sequences. The curves flatten out further out from the origin, indicating that the more likely sequences from PandoGen are also more potent. This trend holds similarly for other methods as well, but the curves flatten pretty early, whereas PandoGen case counts continue to increase for lower-ranked sequences as well. This also shows that the salience of sequences from PandoGen is higher, as the case count increase is not proportional to the increase in the number of novel sequences from PandoGen, but far higher.

In Figures 3, 4, we looked at stats for novel sequence generations from the model. We next look

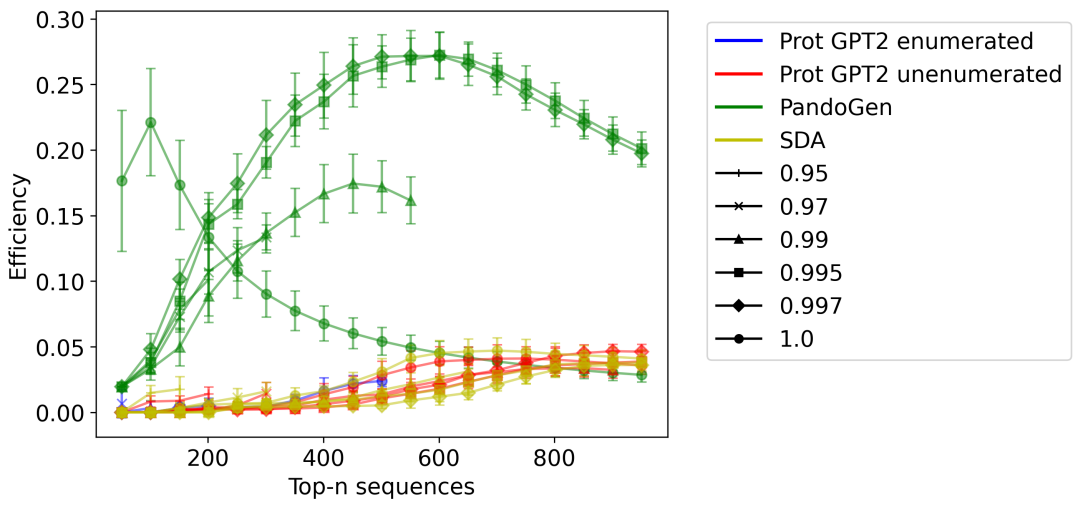

Figure 5: Fraction of real, novel sequences among all sequences generated from the models by sequence rank. Error bars represent the 95% confidence intervals for six independent sampling runs.

at the fraction of novel, real sequences produced from the models among *all* sequences generated from the models in Figure 5. This plot indicates that since other methods are trained on the training set, they are not able to produce novel, real sequences at as high a rate as PandoGen. During PandoGen training, we explicitly forced the model to learn to generate sequences not in the training set, and added guardrails to prevent it from deviating so far from the training set that the outcome is outright wrong. This results in a larger fraction of novel, real sequences in the output samples from PandoGen.

We note based on Figure 2 that the sample differences of PandoGen's outputs are higher than that of competing methods. The question naturally arises as to whether the improvement in PandoGen's performance is purely due to its access to operating points that are not available to the other models. It is possible to expand the operating range of deep autoregressive models through temperature-shaping of the output distribution [AHS85], which has been employed widely in literature [FG17] [CCF+18] [LCC+22]. Adopting temperature shaping, we produce sequences from Prot GPT2 with similar sample distances as PandoGen's. The results indicate that PandoGen results continue to be dominant. These results are presented in the Supplementary Document.

We next examine how many weeks after training the models, are novel sequences from the models first reported in GISAID. We want to look at the trends in novel sequences generated over time, as well as the trends in novel sequences with non-trivial case counts over time. As mentioned in Section 1, sequences with 10 or more cases constitute less than 10% of all sequences. For the purposes of these comparisons we designate sequences which have a total of 10 or more cases in GISAID as "salient" sequences. We also want to track novel Pango lineages [RHO+20] sequences from the models belong to. Pango lineages annotate SARS-CoV2 sequences with lineage labels tracking evolutionary developments in the SARS-CoV2 virus based on phylogenetic analysis. The scheme is designed to track and identify lineages that contribute to the spread of the disease. Pango lineage designation is a widely accepted way to track the SARS-CoV2 virus. New pango lineages indicate potential changes in local and regional epidemiology and represent potential new information about the pandemic. As such, predicting sequences from new lineages ahead of time is a very desirable property for models such as those discussed in this article.

To prepare these comparisons, we selected sampling configurations for PandoGen and Prot GPT2 that resulted in the highest number of novel sequences from all experiments. We ran three sampling experiments with these configurations generating 16,384 sequences in each sampling run.

Figure 6 compares novel sequences generated from the models over time. Both models generate sequences that are reported many weeks in the future, with PandoGen generating a much larger number and continuing to report new sequences well into week 28 after training. For both methods, salient predictions saturate after approximately week 13 after the training period, with PandoGen producing a much larger number of salient sequences than Prot GPT2. The plots also reveal that a large fraction of sequences from PandoGen are salient, whereas a much smaller fraction of sequences from Prot GPT2

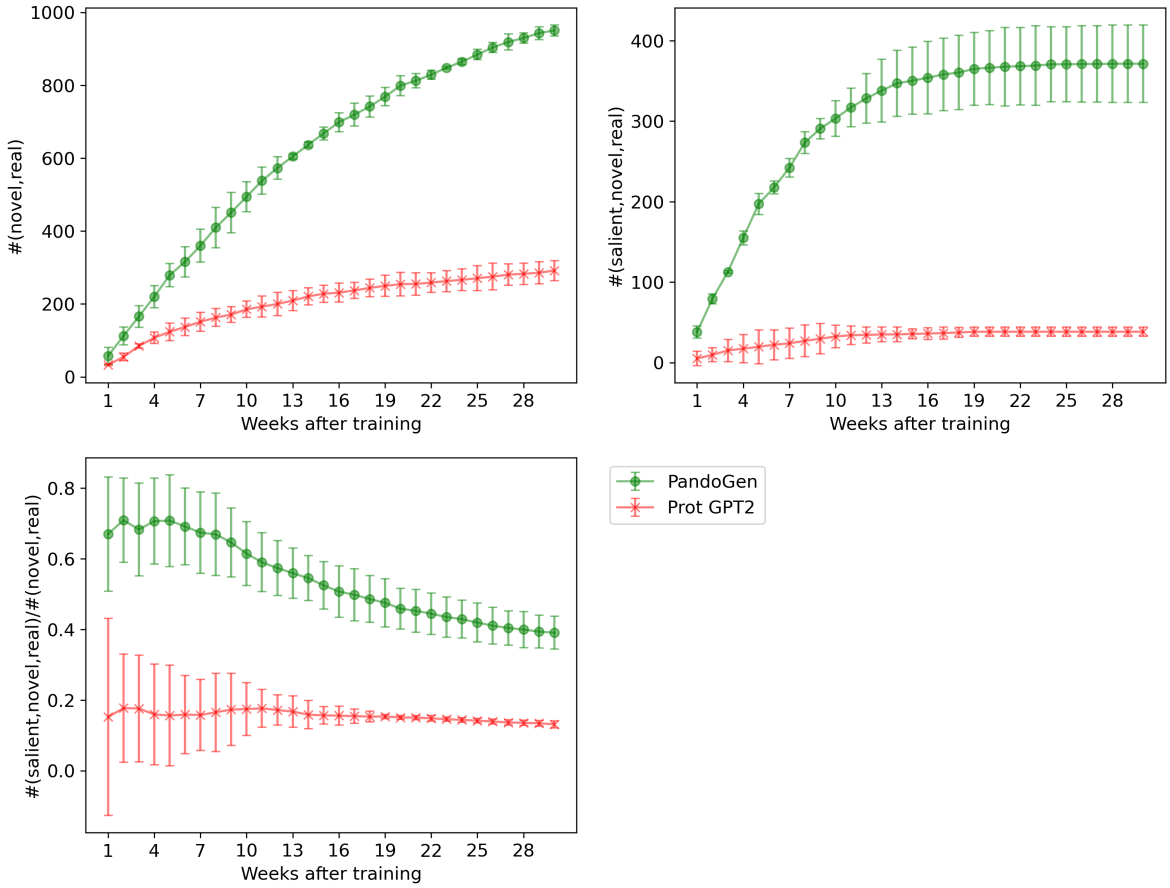

Figure 6: Number of sequences generated within n weeks after the training period. In clockwise direction from the top-left: the first plot represents the number of novel, real sequences reported within n weeks after the training period, the second plot represents the number of novel, real, salient sequences within n weeks after the training period, and the third plot represents the fraction of salient sequences as a fraction of novel predictions from the models. Error bars are from three independent sampling runs, and represent 95% C.I.

are salient. This validates our training approach of using a reward model to train the generative model to produce sequences with high potential for infectiousness.

Figure 7 compares the number of novel lineages in the models' predictions to each other, and to the number of all novel lineages reported in the weeks following the training period. From the figure, Prot GPT2 predicts almost no new lineages, whereas PandoGen predicts tens of novel lineages. The number of new lineages from PandoGen saturates after approximately week 13. It may also be seen that PandoGen predicts a significant proportion of all novel lineages reported for many weeks after training.

Finally, finetuning methods for transfer learning may use the early stopping condition to stop training earlier. We redid the Prot GPT2 experiments where the best checkpoint is chosen through early stopping. For "Prot GPT2 enumerated" case, the selected checkpoint was the same as that presented before. For "Prot GPT2 unenumerated" case, the early stopping checkpoint is from much earlier in the training process. Comparisons between PandoGen and "Prot GPT2 unenumerated" case are in the Supplementary Document. The results presented in the article for Prot GPT2 are seen to be marginally better than the early stopping results in the Supplementary Document.

2.2 Generating instances of notable variants

Next, the ability of PandoGen to generate important Pandemic variants ahead of time, is examined. Since 2022, various subvariants of the Omicron lineage have practically replaced all other variants

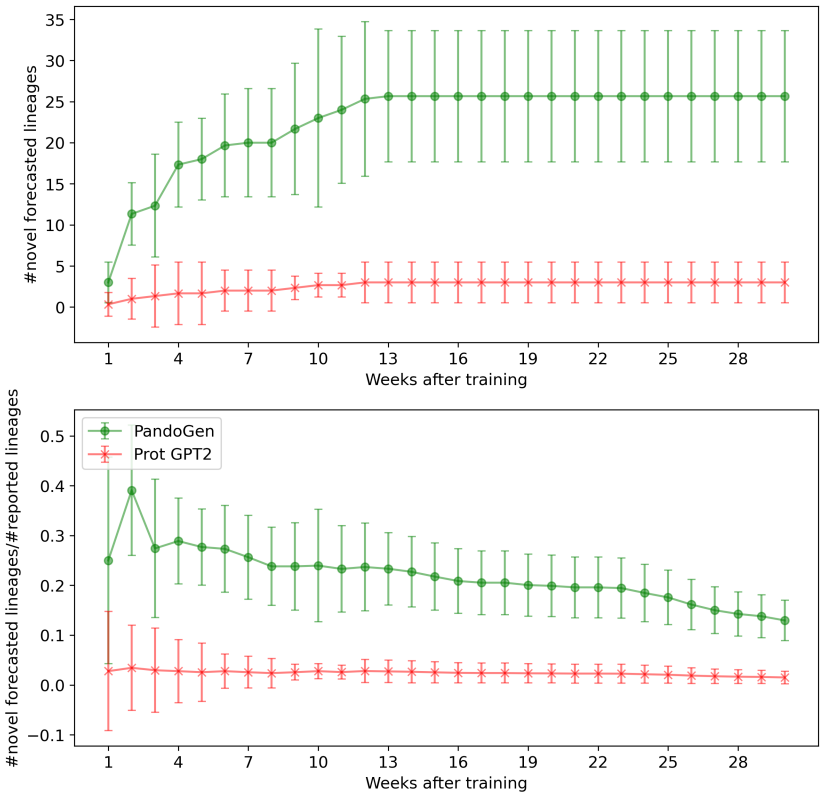

Figure 7: Number of novel lineages in PandoGen predictions vs Prot GPT2. The figure above indicates the raw number of new lineages, and the figure below indicates the new lineages in the samples produced by the models as a fraction of all novel lineages in the database in the same period. Error bars represent 95% C.I.

of SARS-CoV2 in circulation and have become the dominant strains of the virus in circulation by far. The BA.5 subvariant of Omicron was the dominant lineage in circulation since mid- to end-2022, which was then replaced by the BQ.1 subvariant [Par23]. Currently, the XBB.1.5 variant is the dominant Variant of Concern (XBB.1.5 is the lineage with the highest number of sequences submitted to GISAID for each of the last four weeks ending 2023-6-3). Since these are the latest dominant lineages of the virus, experiments were performed targeting these. Prior to the Omicron lineage, the Delta variant (B.1.617.2 and AY.* lineages) caused a large-scale outbreak during the pandemic. To examine PandoGen's performance during the earlier phases of the pandemic, where fewer sequences are available to train on, Delta was also included in the experiments.

For each variant considered, PandoGen was trained using sequences reported prior to the date on which the first sequence belonging to the given variant was reported in GISAID. To determine the first reported date of a variant, we searched the GISAID database for the corresponding lineage(s) or sub-lineages and took the first Spike protein entry with at least one of the mutations characteristic of the variant. For example, for Delta, we looked for all sequences with lineages matching B.1.617.2 or AY.*, and looked for the first reported sequence with one of the three mutations: T478K, P681R, L452R. We referred prior works for mutations in Delta (B.1.617.2 or AY.*) [DSP+22], BA.5 [CBN+23], BQ.1 [AID23], and XBB.1.5 [AHHW23] variants. More details are in Table 1. In these experiments, the training cutoff date is between 10 days to a month before the first reported date for these variants in GISAID.

In the intended use-case sequences generated from PandoGen are to be examined in a laboratory setting through methods such as DMS. Hence only a limited sequence budget is assumed. We assume a sequence budget of $\sim 10^5$ sequences, following the budget of full-sequence DMS experiments [FF14]. To capture a wide range of sequences PandoGen was operated under 11 different nucleus sampling settings with $p \in [0.99, 1.0]$ with a step size of 0.01. For $0.99 \leq p \leq 0.995$ $16,384$ sequences were

generated each, and 4096 sequences for the rest of the settings. This is because, at lower values of p, the model generates less variety as discussed before. While the total number of sequences generated may exceed the sequence budget, the number of novel and unique sequences does not, as models generate the same sequence multiple times. Since these are the only sequences that will be tested using DMS, the sequence budget would be honored.

It is also important to see how consistently PandoGen generates a given variant ahead of time. Hence the sampling procedure was repeated many times for each variant. Success is determined by the ability of the model to generate sequences belonging to the targeted lineage or one of its sublineages, as reported in GISAID. In addition to checking the lineage of the generated sequences, we also check whether the sequences contain at least one characteristic mutation belonging to the variant.

Table 1: Characteristic mutations searched for variants in the experiments

VoC	Characteristic mutations
B.1.617.2, AY.*	T478K,P681R,L452R [DSP+22]
BA.5	G339, S371F, S373P, S375F, T376A, D405A, R408S, K417N, N440K, L452Q, S477N, T478K, E484A, Q493R, Q498R, N501Y, Y505H [CBN+23]
BQ.1	K444T, L452R, N460K, F486V [AID23]
XBB.1.5	V83A, H146Q, Q183E, V213E, G339H, R346T, L368I, F486S, F490S, G252V, F486P [AHHW23]

Table 2: Efficacy of generating variants ahead of time using PandoGen (Spreads represent 95% C.I.)

VoC	Report date	Training cutoff	#Exp	#Gen	#Success
B.1.617.2, AY.*	2021-03-23	2021-02-18	5	11685.6 ± 100.66	5/5
BA.5	2022-03-15	2022-03-05	6	33599.33 ± 135.77	6/6
BQ.1	2022-08-08	2022-07-08	6	36401.83 ± 47.42	5/6
XBB.1.5	2022-10-10	2022-09-30	0	N/A	Failed

The results of the experiments are summarized in Table 2. Except for XBB.1.5, the experiments successfully predicted sequences belonging to the targeted variants ahead of time, and within the requisite sequence budget. For Delta, and BA.5, the predictions are successful in all the experiments. For BQ.1, predictions are successful in 5/6 cases. A higher rate of success may be expected for BQ.1 by increasing the number of sequences generated in this case, as the sequence budget has not been hit in any of these cases.

XBB.1.5 is a sublineage of XBB, a recombinant of two other Omicron lineages. The reward model training was unsuccessful, resulting in a model whose performance was similar to that of random coin-toss for sequences within the training period for XBB.1.5. As a result, the PandoGen pipeline is designated as "failed" for XBB.1.5. This may be because the recombinant parent of XBB.1.5 had been reported only a few weeks before XBB.1.5 itself. In this case, the recombinant breakpoint is located in the Spike sequence's receptor binding domain [TIU+23]. The recombinant event does not represent an accumulation of mutations over time such as in the case of most other notable variants of the virus, but the fusion of two other lineages. Hence, from a sequence prediction point of view, this may present some problems for our model, which is based on training data where such events are rare or not notable in effect [TIU+23].

3 Related Works

There exist many analyses and characterizations of SARS-CoV2 sequences using mathematical models and Machine Learning. These methods offer valuable insights with predictive potential. For instance, in a previous study [HZBB21], the authors presented a method to predict point mutations in the Spike protein with potential for viral escape using machine learning techniques from Natural Language Processing (NLP). Mutations in the Spike protein have been characterized on their effect on binding between the ACE-2 receptor and the Spike protein [LZ22], drawing useful insights connecting ACE-2 binding propensity to the potency of the virus. Deep Mutational Learning [TWG+22], uses machine learning and deep sequencing to produce models that can predict the effect of one or more mutations in the receptor binding module (RBM) of the Spike protein to its binding affinity to both the human ACE-2 receptor as well as to antibodies, offering a detailed picture of the potency of the mutations. In another work [MBW+22], the authors used multiple features including epidemiological, evolutionary, and neural-network-based features to predict the potential of existing individual mutations to become drivers of future Variants of Concern (VoCs). Pyro [OJB+22] is a tool that applies multivariate logistic regression on circulating SARS-CoV2 lineages and their prevalence characteristics, to continually forecast the growth of recent lineages, and predict important mutations with public health implications in existing sequencing data. These studies lay emphasis on evaluation of the effects of mutations, rather than forecasting novel, complete, instances of the Spike protein sequence, which is the focus of this article.

Two classes of PLMs exist in literature today. One class, called Masked Protein Language Models (MPLMs) are trained to predict a set of masked locations in a sequence. This allows the models to learn structural properties of the sequence [RMS+19], and can be finetuned for specific applications using application-specific datasets. Popular ways to use masked models for inferencing are to use the embeddings from the model as features for downstream applications, or to directly finetune the model for classification purposes. During inferencing with the model, it is possible to mask certain locations in a sequence and ask the model to predict the missing locations. In this way, theoretically, all possible point mutations and combinations thereof can be explored using the model. However, it is infeasible in practice to use it as a generative model because (1) only point mutations can be modeled whereas the space of sequences contain insertions and deletions as well and (2) the number of possible masks of a sequence scales exponentially as the length of the sequence, resulting in prohibitive computational costs. Notwithstanding these limitations, ProtFound [CNW+22] tries to model a specific segment within the Spike protein using an MPLM involving the generation of millions of masked sequences. Due to the computationally heavy nature, the pipeline uses a supercomputing cluster to run. The work also doesn't treat the efficacy aspect - as in how many of the sequences produced this way are realistic or found in nature. To scale to the length of the full Spike sequence instead of the segment treated in ProtFound [CNW+22] would need tremendous resources and is likely infeasible as the full Spike sequence is much larger than the segment modeled in that work. Our method solves the problem of full Spike sequence generation on a single compute node within a limited sequence budget.

The second class of PLMs is the type that is used in this work, the deep autoregressive models that were discussed in Section 1. These are generative models of sequences, and can be used as sampling beds for *de novo* sequence generation, without the restrictions faced by MPLMs. These models are usually initially pre-trained on large corpora of unlabeled datasets through simple likelihood maximization methods, and further customized for a specific task through transfer learning [RSR+20] by finetuning the model on a task-specific dataset also through likelihood maximization. As detailed before there exist multiple examples of deep autoregressive models used to model biological sequences [MKG+23] [FSH22] [SRK+21]. These methods are all trained, and in two cases, finetuned [MKG+23] [Dho23], using standard likelihood maximization. As outlined in Section 1, this approach has many limitations when it comes to generating sequences in a pandemic situation, and our goal is to introduce training methods to counter these issues to some extend.

Related to our work are methods in Natural Language Processing such as Reinforcement Learning from Human Feedback [CLB+23]. Recently, a method called Quark [LWJ+22] was proposed as a drop-in replacement for Reinforcement Learning for the same class of problems. Quark was originally intended to reduce toxicity in generated text using human labels for LLMs that respond to human input. We modify and adapt the solution structure of Quark for performing model updates within the PandoGen training loop as shown in Figure 1.

4 Methods

4.1 SARS-CoV2 sequences

We downloaded the file variant_surveillance.tsv from GISAID, which contains sequence and metadata (submission date, Pango Lineage, list of mutations, geographic location etc). From this file, for each entry, we extracted mutations in the Spike protein sequence, and applied them to the GISAID-designated Spike protein reference to obtain the actual Spike sequences. We collected all sequences except those containing a stop codon mutation, as premature stop codons have a propensity to result in non-functional proteins [DBBZ18]. The experiments in this article were performed over a period of time, and we used two different releases of variant_surveillance.tsv files from GISAID. All experiments in Section 2.1 as well as experiments in Section 2.2 for the Delta variant and Omicron subvariants BA.5 and BQ.1 were performed using a GISAID release in which the last sequence was submitted on 2022-11-29. The Accession IDs of sequences in this file have been deposited in GISAID with the following EPISET ID: EPI_SET_230807ta. For the XBB.1.5 experiment in Section 2.2 we used a more recent release from GISAID in which the last sequence was submitted on 2023-06-03. The Accession IDs of sequences in this file have been deposited in GISAID with the following EPISET IDs: EPI_SET_230809ud, EPI_SET_230809pa.

For comparing sequences, we allow ambiguous amino acid characters to match with their corresponding disambiguated targets. That is, we consider "B" to be equal to either "D" or "N", "J" to be equal to either "L" or "I", "Z" to be either "Q" or "E", and "X" to be any amino acid. The matching algorithm, in full detail, is given in the Supplementary Document. Sequences are first checked to see whether they match sequences reported in the training period. Sequences not matching sequences in the training period are designated as novel sequences. Novel sequences are checked among sequences first reported after the training period to determine which novel sequences are real sequences.

For reporting the number of generated sequences in Section 2.2, where only the order of magnitude of unique sequences is relevant, we used the computationally inexpensive exact string matching predefined in Python.

4.2 Model architecture

We base our models on decoder-only autoregressive models, which consist of stacks of modules, each module consisting of a self-attention layer, a layer-normalization layer, a non-linearity and dropout layer [VSP+17]. The model has eight of these modules, with attention head dimensionality of 128, with 12 attention heads, and a fully connected layer that is three times the attention size. On top of the 8 stacked modules, we have a final Softmax layer that predicts a distribution over the protein sequence elements (amino acids). We call this final layer, the Protein Model head (PM head), for ease of reference later. Overall, there are approximately 192 million parameters in our models (SDA and PandoGen).

Given a training set, each amino acid character in a protein sequence in the training set is converted to an embedding; hence the input protein sequence becomes a sequence of embeddings of the same length. To this sequence, we prepend a special character indicating start of sequence. To this, a second embedding sequence of the same length and dimensionality is added. This second embedding indicates the position of each embedding in the sequence, and is trained along with the rest of the model. These conventions largely follow standard Transformer models [VSP+17].

The model predicts the probability of a protein sequence as follows.

$$\log P[X_{1:N}] = \sum_i \log P(x_i \mid X_{1:i-1}; \Phi) \tag{6}$$

Here, Φ represents the parameters of the model, and $P(x_i \mid X_{1:i-1}; \Phi)$ is the output of the PM head. During data generation from the model, the generation is conditioned on the start of sequence character, that is, we ask the model to complete the sequence provided only the start of sequence special character.

4.3 Pretraining on UniProt dataset

We downloaded the UniProt version $2019 - 09$ which contains protein sequences released before the onset of the COVID-19 pandemic. We first pretrained our 192M parameter model using UniRef50 sequences, split randomly into 35M training sequences and 3.9M validation sequences. Pretraining was done for our models using snippets of UniRef50 sequences of length 255. Where a sequence was shorter than length 255, the complete sequence was used and when sequence was longer, a random substring of length 255 was sliced out. During validation, to keep the validation set deterministic, we used only sequences shorter than length 256. We trained the model for two epochs with a learning rate of 1×10^{-4}, and a weight decay of 1×10^{-2}. We employed learning rate warmup for 1024 batches, and thereafter, a linear decay in learning rate was instituted. We used 4 NVIDIA V100 GPUs to pretrain the model using half-precision floating point with a batch-size of 30 per device. Validation loss was calculated once every quarter of an epoch, and the model version with the best validation loss was retained for further finetuning. We call this model, the UniProt Deep autoregressive Model (UDA), for ease of reference.

4.4 Finetuning on SARS-CoV2 Spike protein sequences

GISAID datasets were created as described in Section 4.1. To create the training dataset from GISAID, we selected a cutoff date, and sequences on or before this cutoff date were used for training. The goal would be to predict sequences after this cutoff date.

The sequences were split into training and validation sequences based on a random split of the Pango lineages [RHO+20] assigned to the sequences. Any Spike sequence occurring in both a training and validation pango lineages was moved to the training set.

We finetuned the model resulting from Section 4.3 as well as the Prot GPT2 model, on this same training/validation split, both for 24 epochs, checkpointing the models four times in each epoch. Both models used a fine-tuning learning rate of 1×10^{-5}, with a weight decay of 1×10^{-2}, 1024 learning rate linear warmup steps and linear decay afterwards. We evaluated each checkpoint on the validation set, and selected the one with the best loss.

We term the finetuned version of our UDA model as the SARS-CoV2 deep autoregressive model (or SDA model) in the sequel, and the Prot GPT2 finetuned model is referred to as "Prot GPT2 (unenumerated)" as the model is finetuned on a unique set of SARS-CoV2 sequences without information about prevalences of different sequences. Both are from a similar family of models, and expected to have similar performance characteristics. However Prot GPT2 is a much larger model with close to 800 million parameters.

In addition, we trained "Prot GPT2 (enumerated)" where the training sequences are not made unique. So, if a sequence is deposited in GISAID 500 times, the sequence occurs in the training set 500 times as well. The training data size for this case resulted in a similar number of gradient updates for a single epoch as 24 epochs of the data with unique sequences used for finetuning the other models. Hence training in this case was performed for a single epoch only. AS for the "Prot GPT2 (unenumerated)", model we selected the checkpoint with the best validation loss.

While the primary Prot GPT2 models followed the model selection criterion described above, we also used a second model selection method for Prot GPT2 with early stopping; details of this are presented in the Supplementary Document as mentioned elsewhere, as these second set of models do not perform better than the primary models.

4.5 Reward model training

The reward model's goal is to predict potential infectiousness for an input sequence as a scalar value. We would like to use the number of times a Spike protein sequence has been reported in the GISAID database as a signal indicating the infectiousness of the sequence. However, it is not straightforward to use this data due to the following reasons. First, a sequence discovered earlier during the pandemic has had more time to spread, compared to a newer sequence. Second, sequencing resources may have needed time to ramp up around the world. Third, restrictions and interventions against the pandemic have shown variation over time. As a result of these, using the raw GISAID case counts as a label is misleading. For instance, the original strain of the SARS-CoV2 virus discovered in 2019 has had many

years to circulate, whereas a more recently discovered sequence has had very little time, but their case counts may imply that the original strain is more infectious, which is not necessarily true.

We simulate a game between two sequences. The arena for the game between Seq A and Seq B, is the set of all GISAID cases within the training period that belong to Seq A and Seq B. The game is to randomly select an item from this set. Seq A wins if the selected item is Seq A, and Seq B wins otherwise. The probability of Seq A winning a game is $\frac{N_A}{N_A+N_B}$ where N_A, N_B are the GISAID case counts of Seq A and Seq B respectively.

We ensure that the games are played only between sequences whose first reported date in GISAID is within a narrow timeframe of each other. Implicit here is the assumption that this ensures that they were under similar constraints throughout their lifetimes within the training period, and that their case counts are sufficiently indicative of their relative infectiousness. Sequences can have a small difference between their respective discovery dates (discovery dates are dates on which the sequences are first reported in GISAID), depending on how clear, on average, the winner is, between the two sequences. The largest difference in sequence discovery date in our dataset is 5 weeks, but this requires one sequence to have at least 10^5 times more cases than the other. As the ratio of case counts between sequences shrinks, the allowed gap between their discovery dates is also tightened. The scheme is detailed in Table 3. As shown, as the case count ratio between two sequences increases, they are allowed to be discovered further apart in time to be inducted into the competition draw. We also do not use any sequences first reported in the last 7 weeks of the training period (sequences reported within the 7 week period, which were first reported before that period, are still considered), as the sequences discovered in this time frame may not have had sufficient time to spread.

Examples of considerations when creating the competition draw are illustrated in Figure 8. When sequences are discovered close to each other in time and they have a large difference in case counts, the competition between the sequences are accepted. Small differences in discovery dates are not deemed to upset who the winner between the sequences is in the majority of games due to the overwhelming number of cases of one sequence compared to the other. An example of this is the Sequence A vs Sequence B competition. When sequences have very similar case counts, the requirements are more stringent. In these cases, (1) the sequences would need to be discovered in the same week, (2) their relative case counts must be similar in two disjoint geographic regions, and (3) the winner in the majority of the games must be the same in the two geographic regions. Sequence C vs Sequence A is rejected because they have almost the same case counts, but were discovered 2 weeks apart. Sequence D vs Sequence E will be considered a valid competition if their relative case counts are similar in two disjoint geographic regions and Sequence D is the winner in both regions.

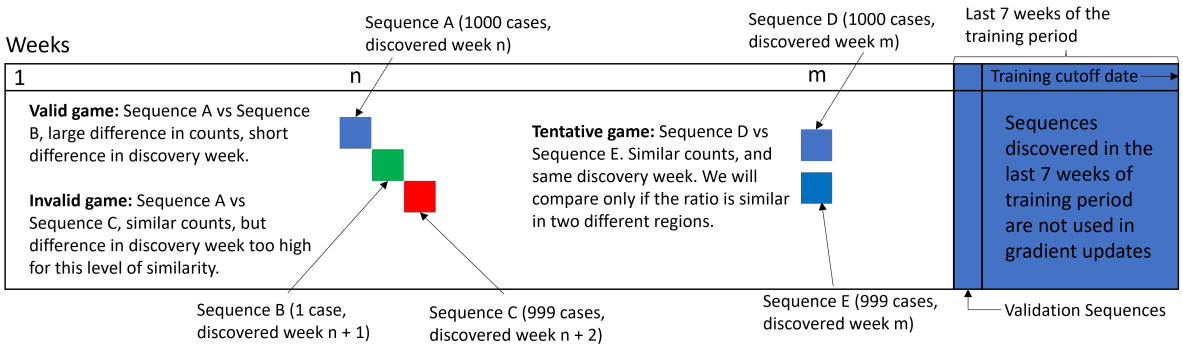

Figure 8: Heuristics for creating *fair* games. We allow games only among sequences first reported within a certain timeframe of each other.

We apply additional heuristics so that the number of comparisons are tractable. Each sequence can be compared to a maximum of 750 sequences in competitions (random subsampling is used to cut down larger lists). A maximum of 2,000,000 sequences are randomly sampled from the resultant draw. After this, we run some filtrations. First, sequence pairings whose majority winner changes in two disjoint geographical regions (Eurasia vs non-Eurasia) are removed. Second, sequence pairs where the majority winner's win proportion differs in the two separate geographic region by more than 0.25 are discarded as well. Finally, if the total number of surviving competitions exceeds 1,000,000 competitions, we subsample the list to 1,000,000 cases. The case counts of two competing sequences

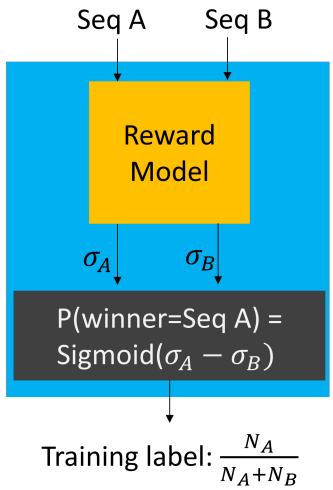

Figure 9: Reward model training method. The reward model processes two incoming sequences Seq A and Seq B, and produces infectiousness potentials σ_A, σ_B for these respectively. These are converted to probabilities for competition outcomes in a differentiable manner. The probabilities are trained on the ground-truth label. The probability of Seq A winning is trained on the label value $\frac{N_A}{N_A+N_B}$ where N_A, N_B are GISAID case counts of the two sequences within the training period.

Table 3: Ratio of case counts of the lower-occurrence sequence to that of the higher-occurrence sequence and allowed tolerance in their weekly discovery dates for them to be allowed to be compared

Case count ratio (r)	Discovery week tolerance
$0 \leq r < 1e-5$	5
$1e-5 \leq r < 1e-4$	4
$1e-4 \leq r < 1e-3$	3
$1e-3 \leq r < 1e-2$	2
$1e-2 \leq r < 1e-1$	1
$1e-1 \leq r < 1$	0

in a pair is taken to be the cumulative case count for the first l weeks where l is the minimum of the number of weeks of the two sequences' lifetimes within the training period.

We use this data to train the reward model (Figure 9). During training, we fetch paired sequences from the competition draw. Lets say we selected Seq A, and Seq B. The reward model predicts an infectiousness potential for Seq A, and an infectiousness potential for Seq B, which are denoted as values σ_A, σ_B respectively. The probability distribution of the outcome is predicted as $P(Seq\ A\ wins) = \frac{\exp(\sigma_A)}{\exp(\sigma_A)+\exp(\sigma_B)}$. This predicted distribution is trained using the "actual" distribution which is $P(Seq\ A\ wins) = \frac{N_A}{N_A+N_B}$ by minimizing the KL-divergence between the two. Note that this causes the reward model to learn to predict higher values for σ_A if Seq A is the winner in majority of the games, hence learning to predict a potential for each sequence that increases with the infectiousness potential of the sequence.

The reward model uses the same architecture as the SDA model, except that the PM head is replaced with a reward head. A PM head produces as many outputs as there are characters in our amino acid alphabet, whereas the reward head produces a single scalar output. The rest of the layers in the reward model are initialized from the SDA model's parameter values. Also, the reward head output is only valid after the complete sequence has been passed through the model.

The reward model was trained with a learning rate of 1×10^{-5} over 1 epoch. Learning rate warmup was instituted for the first 1024 steps, and then linearly decayed. Weight decay of 10^{-2} was also used. A total of four checkpoints were kept during training. For selecting the checkpoint for downstream use, we used sequences discovered in the 7th week from the end of the training period (note that these

sequences were not used in training; see Figure 8). Among these, sequences having an occurrence count of over 50 in the last 7 weeks of the training period were labeled 1 and the remaining sequences were labeled 0. The sequences were each scored by the checkpoints, and the area under the receiver operating characteristic (AUROC) curve, a popular metric for model selection for binary classifiers, was computed. The checkpoint performing the best was chosen.

4.6 PandoGen finetuning

The final step in our pipeline is to finetune the SDA model directly within its operational setting of sequence generation, using the reward model we presented in Section 4.5. To finetune the model, in addition to the reward signal, we also use a second signal indicating whether a sequence generated by the model already exists in the list of known sequences or not.

To perform this finetuning, we adapt an algorithm called Quark [LWJ+22]. Quark was developed to improve an LLM's ability to generate text that conforms to certain qualities such as reduced toxicity. We prefer Quark to Reinforcement Learning [CLB+23] [SOW+20] for model updates because it has been reported to be stabler, and easier to implement using libraries created for LLM training.

To explain PandoGen training, first we recollect (Section 4.2) that the SDA model generates sequences conditioned on a special beginning of sequence character. The PandoGen model, which is a finetuned version of the SDA model, generates sequences conditioned also on a second special character. This second special character corresponds to sequence quality. We call this, the reward token. During training, we generate sequences from PandoGen conditioned on the *highest quality reward token* (to be explained in a moment). The generated sequences are scored using the reward model, and then looked up in the set of historically known sequences. If the sequence is historically known, we assign it a special reward token indicating historicity. If not, we assign it the reward token associated with the quantile within which the reward for the generated sequence lies. The quantiles are determined based on generations from the SDA model scored by the reward model. Hence, the highest quality reward token is assigned to sequences with rewards in the highest quantile, but which are not sequences in the original training set. The generated sequences annotated with the reward tokens are added to a data pool, $\mathcal{D}$ of sequences (initialized using data generated from the SDA model). A fixed-size sample, D, is taken from this data pool, and used to perform a finetuning step. The finetuning step maximizes the following objective through mini-batch gradient descent.

$$E_{X\sim D}\sum_i \Big\{ \log P[x_i \mid X_{1:i-1}, r_X; \Phi] - \beta D_{KL}[p(x_i \mid X_{1:i-1}; \Phi_{SDA}) \| p(x_i \mid X_{1:i-1}, r_X; \Phi)] \Big\}$$

Here, Φ_{SDA} represents the parameter set of the original SDA model which is not adjusted during PandoGen finetuning, and Φ represents the parameter set of the PandoGen model, which is initialized from Φ_{SDA}, and modified through the finetuning step. r_X is the reward token for the sequence X, assigned based on the procedure mentioned above. $D_{KL}(p||q)$ is the KL-divergence between two distributions, p, q. Note that the learning objective here is slightly different from the original Quark cost function, which requires D to be a stratified subset of $\mathcal{D}$, where sequences in each reward quantile are uniformly represented. We also implemented a looser version of stratified sampling as an option for hyperparameter tuning (explained below). In our implementation, we continue sampling from different quantiles in $\mathcal{D}$, and stop when 1) only one quantile is left and 2) the remaining quantile has the highest representation in D.

The procedure of generating sequences from the PandoGen model, scoring it using the reward model, checking historicity, and finetuning the model using the annotated generations is carried out iteratively. Through the procedure, the PandoGen model is trained to generate sequences according to their quality and historicity (via conditioning on the reward token, r_X). It learns to generate high reward, non-historic sequences when conditioned on the highest quality reward token. At the same time, the loss term $\beta D_{KL}(p(x_i \mid X_{1:i-1}; \Phi_{SDA}) \| p(x_i \mid X_{1:i-1}, r_X; \Phi))$ prevents the model from deviating too far from the probability distribution learnt by the original SDA model. This is important as the SDA model has a grasp of the structure of the SARS-CoV2 Spike protein sequence, and deviating too far from the basic structure is detrimental to our goals. The training scheme is shown in Figure 10.

For PandoGen finetuning we perform a lazy version of hyperparameter tuning to save compute resources and time. We run PandoGen finetuning steps with default hyperparameter settings for 24

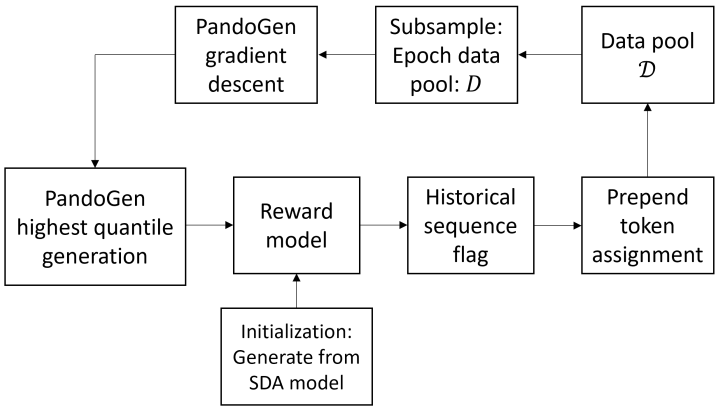

Figure 10: The PandoGen finetuning procedure follows three steps: sequence generation, sequence scoring/categorization, and finetuning using the scored/categorized sequences

epochs. The default hyperparameter case involves not using dropout, and not using stratified sampling. These settings were selected based on small-scale experiments. We use the early stopping criterion to terminate finetuning, whereby if the rewards of the generated batches (evaluated by the reward model) do not increase for three consecutive epochs, the training is halted. When training is halted, we examine the average reward for a batch generated from the best model checkpoint. We require that this reward be in the top two quantiles of rewards for samples from the SDA model prior to PandoGen finetuning. If this condition is met, we select the model for experiments. We use this criterion because the PandoGen finetuning method, based on Quark, teaches the model to generate sequences from the highest reward quantile. If the model's performance approaches this training objective, we do not expect further appreciable gains by changing hyperparameters. On the other hand if the goal is not met, we deem the training not to have succeeded, because the training objective (generating high quantile sequences) is not met. In this case, we launch hyperparameter search. The following hyperparameter settings were explored in this case: 1) using stratified sampling, 2) no early stopping (train for all 24 epochs and select the best), and 3) train with stratified sampling and dropout. The best model checkpoint (as determined by the reward model) from these three runs is chosen for downstream experiments. Note that the model-selection method used here does not use data outside of the training period, and simply uses the reward model to determine the right model checkpoint and test convergence. The only case where the default training hyper-parameters yielded a model which did not meet the model selection criteria was for the Delta variant experiment in Section 2.2. For this case, hyperparameter search revealed that stratified sampling and dropout provided the best results, and this model was used for further downstream experiments and analyses.

5 Discussion

We presented a novel method to train deep generative models to generate future pandemic sequences with the goal that the generated sequences must have a high fraction of real sequences, and that the case count of the generated corpus should be high. We compared our training pipeline with a standard model in the field and found that our method outperforms the existing method by 4 times in terms of number of novel sequences and close to 10 times for case counts of the generated corpus, indicating that alignment of large sequence models to the purposes of generating sequences during a pandemic doesn't automatically happen through transfer learning approaches. Meanwhile, our training pipeline used only information available in GISAID, and did not depend on additional laboratory experiments to perform sequence characterization, making it an attractive approach. We also found that our method is able to predict sequences belonging to important variants before they were available in GISAID.

It is of vital importance that we prepare for the next pandemic. Information-sharing systems such as GISAID were setup very quickly, to combat the COVID-19 pandemic. A system supporting such international data-sharing is vital to analytical efforts such as ours as prediction of future sequences and other important elements of the pandemic can be done more accurately and in a timely manner

if the most up to date information is available. We note that there are delays between sequences discovered in the real world versus when they are available in GISAID. But we hope these delays can be significantly reduced with GISAID or another system in case we find ourselves in the unfortunate situation of having to combat another global pandemic. As sequencing infrastructure improves and becomes more commonplace around the world, we have hopes that this will indeed become the case. As such, we hope methods such as ours, will provide valuable tooling supporting pandemic management in the future.

While we presented promising results, we note that there is potential for further research in the future. Being able to predict recombinant lineages is a valuable next step. Recombinant events between two different sublineages typically need co-infection of the same host [FM22]. Since these events are not as frequent as mutation accumulation events, and since the effect of recombinant events may not be pronounced as frequently, additional considerations will be involved. PandoGen's framework of guiding the generator using the discriminative model can be extended to such cases with some changes. During the training process the generator must generate recombinant candidates and the discriminator must rate them as stable, feasible viral sequences. To bootstrap this process and target it towards recombinants, we may need to resort to data augmentation approaches as naturally occurring sequences are not rich in such scenarios. Data augmentation can be envisaged to include the following steps: (1) identification of candidate lineage pairs that may result in recombinants based on sublineages infecting common regions, (2) generation of candidate recombinants by combining the collected pairs, (3) ranking of recombinant candidates through PandoGen models, and through other approaches that can be used to determine protein fitness [HNFL22], (4) retraining of the PandoGen reward model to favor candidates ranked high against low-ranked ones and the PandoGen generative model to generate high-ranking candidates, and (5) iterative refinement of the PandoGen generator using the PandoGen discriminator. Another potential direction is to integrate PandoGen in a laboratory setting to continually improve the model through incorporation of experimental data. For instance, after PandoGen is trained as in this article, purely using sequence information in GISAID, a second dataset can be prepared where sequences generated from PandoGen are evaluated in the laboratory setting using DMS to annotate them with multiple properties. Given that these new sequences will have rich annotations that directly reflect the phenotypical characteristics of the virus, this offers an avenue to incorporate more accurate and precise information into a second PandoGen training loop offering more potential improvements. Finally, model sizes [NC22], training infrastructure [PyT23], and gradient update algorithms [HSW+21] are evolving constantly, allowing larger models to be finetuned more efficiently. Such developments will allow bigger models to be incorporated into methods such as ours holding promise for improvements continuing into the future.

6 Acknowledgement

This work utilizes resources supported by the National Science Foundation's Major Research Instrumentation program, grant #1725729 [KMZ+20], as well as the University of Illinois at Urbana-Champaign.

We gratefully acknowledge all data contributors, i.e., the Authors and their Originating laboratories responsible for obtaining the specimens, and their Submitting laboratories for generating the genetic sequence and metadata and sharing via the GISAID Initiative, on which this research is based.

7 Author Contribution

AR designed and implemented the methods and performed the experiments for this article. SSL contributed to the design of the PandoGen pipeline, participated in the analysis of results, and the preparation of the manuscript. DC directed the complete project including project goals, design, engineering and implementation. All authors have read and approved the submission of manuscript.

8 Competing Interests

The authors declare that they have no competing interests.

9 Materials and Correspondence

Corresponding Author: Deming Chen

10 Data availability

All data used for the study are available at GISAID's EpiCov repository, as well as from the UniRef database.

Accession IDs of GISAID sequences have been deposited in GISAID under the following EPISET IDS: EPI_SET_230809pa, EPI_SET_230809ud, EPI_SET_230807ta.

UniRef50 dataset release is 2019_09.

11 Code availability

The source code for PandoGen is available at https://github.com/UIUC-ChenLab/PandoGen

References

[AHHW23] Danyi Ao, Xuemei He, Weiqi Hong, and Xiawei Wei. The rapid rise of sars-cov-2 omicron subvariants with immune evasion properties: Xbb. 1.5 and bq. 1.1 subvariants. *MedComm*, 4(2):e239, 2023.

[AHS85] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[AID23] Shopnil Akash, Md Rezaul Islam, and Kuldeep Dhama. Emergence bq. 1 and bq. 1.1 as newly identified omicron subvariants: current scenario and future outlook–an update. *Annals of Medicine and Surgery*, 85(4):1329, 2023.

[CBN+23] Srijan Chatterjee, Manojit Bhattacharya, Sagnik Nag, Kuldeep Dhama, and Chiranjib Chakraborty. A detailed overview of sars-cov-2 omicron: its sub-variants, mutations and pathophysiology, clinical characteristics, immunological landscape, immune escape, and therapies. *Viruses*, 15(1):167, 2023.

[CCF+18] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

[CLB+23] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

[CNW+22] Jie Chen, Zhiwei Nie, Yu Wang, Kai Wang, Fan Xu, Zhiheng Hu, Bing Zheng, Zhennan Wang, Guoli Song, Jingyi Zhang, et al. Running ahead of evolution-ai based simulation for predicting future high-risk sars-cov-2 variants. *bioRxiv*, pages 2022–11, 2022.

[DBBZ18] Maciej Dabrowski, Zuzanna Bukowy-Bieryllo, and Ewa Zietkiewicz. Advances in therapeutic use of a drug-stimulated translational readthrough of premature termination codons. *Molecular medicine*, 24:1–15, 2018.

[DCR+23] Bernadeta Dadonaite, Katharine HD Crawford, Caelan E Radford, Ariana G Farrell, C Yu Timothy, William W Hannon, Panpan Zhou, Raiees Andrabi, Dennis R Burton, Lihong Liu, et al. A pseudovirus system enables deep mutational scanning of the full sars-cov-2 spike. *Cell*, 186(6):1263–1278, 2023.

[DEKM98] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[Dho23] Rahul M. Dhodapkar. A deep generative model of the sars-cov-2 spike protein predicts future variants. *bioRxiv*, 2023.

[DSM+21] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*, 2021.

[DSP+22] Manish Dhawan, Abhilasha Sharma, n Priyanka, Nanamika Thakur, Tridib Kumar Rajkhowa, and Om Prakash Choudhary. Delta variant (b. 1.617. 2) of sars-cov-2: Mutations, impact, challenges and possible solutions. *Human Vaccines & Immunotherapeutics*, 18(5):2068883, 2022.

[EHD+21] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

[FF14] Douglas M Fowler and Stanley Fields. Deep mutational scanning: a new style of protein science. *Nature methods*, 11(8):801–807, 2014.

[FG17] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

[FM22] Daniele Focosi and Fabrizio Maggi. Recombination in coronaviruses, with a focus on sars-cov-2. *Viruses*, 14(6):1239, 2022.

[FSH22] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

[HBD+19] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[HMRC21] Yun Heo, Gowthami Manikandan, Anand Ramachandran, and Deming Chen. Comprehensive evaluation of error-correction methodologies for genome sequencing data. *Exon Publications*, pages 89–108, 2021.

[HNFL22] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Learning protein fitness models from evolutionary and assay-labeled data. *Nature biotechnology*, 40(7):1114–1122, 2022.

[HSW+21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[HZBB21] Brian Hie, Ellen D. Zhong, Bonnie Berger, and Bryan Bryson. Learning the language of viral evolution and escape. *Science*, 371(6526):284–288, 2021.

[KMZ+20] Volodymyr Kindratenko, Dawei Mu, Yan Zhan, John Maloney, Sayed Hadi Hashemi, Benjamin Rabe, Ke Xu, Roy Campbell, Jian Peng, and William Gropp. Hal: Computer system for scalable deep learning. In *Practice and experience in advanced research computing*, pages 41–48. 2020.

[LCC+22] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[LWJ+22] Ximing Lu, Sean Welleck, Liwei Jiang, Jack Hessel, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. Quark: Controllable text generation with reinforced unlearning. *arXiv preprint arXiv:2205.13636*, 2022.

[LZ22] Zhendong Li and John ZH Zhang. Mutational effect of some major covid-19 variants on binding of the s protein to ace2. *Biomolecules*, 12(4):572, 2022.

[MBW+22] M. Cyrus Maher, Istvan Bartha, Steven Weaver, Julia di Iulio, Elena Ferri, Leah Soriaga, Florian A. Lempp, Brian L. Hie, Bryan Bryson, Bonnie Berger, David L. Robertson, Gyorgy Snell, Davide Corti, Herbert W. Virgin, Sergei L. Kosakovsky Pond, and Amalio Telenti. Predicting the mutational drivers of future sars-cov-2 variants of concern. *Science Translational Medicine*, 14(633):eabk3445, 2022.

[MKG+23] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pages 1–8, 2023.

[NC22] Sharan Narang and Aakanksha Chowdhery. Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance, 2022.

[ODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[OJB+22] Fritz Obermeyer, Martin Jankowiak, Nikolaos Barkas, Stephen F Schaffner, Jesse D Pyle, Leonid Yurkovetskiy, Matteo Bosso, Daniel J Park, Mehrtash Babadi, Bronwyn L MacInnis, et al. Analysis of 6.4 million sars-cov-2 genomes identifies mutations associated with fitness. *Science*, 376(6599):1327–1332, 2022.

[Ope23] OpenAI. Gpt-4 technical report, 2023.

[OWJ+22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[Par23] Dinah V Parums. The xbb. 1.5 ('kraken') subvariant of omicron sars-cov-2 and its rapid global spread. *Medical Science Monitor: International Medical Journal of Experimental and Clinical Research*, 29:e939580–1, 2023.

[PyT23] Team PyTorch. Pytorch 2.0: Our next generation release that is faster, more pythonic and dynamic as ever, 2023.

[RHO+20] Andrew Rambaut, Edward C Holmes, Áine O'Toole, Verity Hill, John T McCrone, Christopher Ruis, Louis du Plessis, and Oliver G Pybus. A dynamic nomenclature proposal for sars-cov-2 lineages to assist genomic epidemiology. *Nature microbiology*, 5(11):1403–1407, 2020.

[RMS+19] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019.

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[SM17] Yuelong Shu and John McCauley. Gisaid: from vision to reality. *Eurosurveillance*, 22(13):30494, 2017.

[SOW+20] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[SRK+21] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):2403, 2021.

[TIU+23] Tomokazu Tamura, Jumpei Ito, Keiya Uriu, Jiri Zahradnik, Izumi Kida, Yuki Anraku, Hesham Nasser, Maya Shofa, Yoshitaka Oda, Spyros Lytras, et al. Virological characteristics of the sars-cov-2 xbb variant derived from recombination of two omicron subvariants. *Nature communications*, 14(1):2800, 2023.

[TWG+22] Joseph M Taft, Cédric R Weber, Beichen Gao, Roy A Ehling, Jiami Han, Lester Frei, Sean W Metcalfe, Max D Overath, Alexander Yermanos, William Kelton, et al. Deep mutational learning predicts ace2 binding and antibody escape to combinatorial mutations in the sars-cov-2 receptor-binding domain. *Cell*, 185(21):4008–4022, 2022.

[uni17] Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 45(D1):D158–D169, 2017.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Xia21] Xuhua Xia. Domains and functions of spike protein in sars-cov-2 in the context of vaccine design. *Viruses*, 2021.