Heidelberg University
Institute of Computer Science

Project report for the lecture Advanced Machine Learning

Prediction of the next SARS-CoV-2 variants

https://github.com/nilskre/AML-covid-project

Team Member: Felix Hausberger, 3661293,
Applied Computer Science
eb260@stud.uni-heidelberg.de

Team Member: Nils Krehl, 3664130,
Applied Computer Science
pu268@stud.uni-heidelberg.de

Plagiarism statement

We certify that this report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication.

We also certify that this report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.

Member contributions

Nils Krehl

Felix Hausberger

Contents

0 Project Setup **2**

1 Introduction **2**

2 Fundamentals and Related Work **3**
2.1 Basics from molecular biology 3
2.1.1 Genome sequences 3
2.1.2 Protein biosynthesis 3
2.1.3 Structure of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) 4
2.1.4 Relationships between biological objects 5
2.2 From Probabilistic Language Models to modeling Evolution Theory 6
2.3 GISAID EpiFlu Data Platform 9
2.4 Domain-Specific Methodologies to create Evolutionary Datasets for Mutation Prediction 9
2.5 Previous Work on Mutation Prediction 11
2.6 Sequence to Sequence Models based on Long Short-Term Memory 12
2.7 Applying Generative Adversarial Networks 13
2.8 Transformer and Attention Mechanism 15
2.9 Other Techniques 18

3 Approach **19**
3.1 Dataset Creation 19
3.1.1 Raw data selection from GISAID 19
3.1.2 Generation of a phylogenetic tree 19
3.1.3 Phylogenetic tree to dataset 19
3.2 Data Preprocessing 19
3.2.1 Dimensionality reduction by selecting subpart of the genome 19
3.2.2 Transform genome sequence to numeric model input . . 19
3.3 Model Architecture 19
3.4 Training Process 19

4 Experimental results **20**

5 Conclusion **21**

List of Abbreviations

DNA	Deoxyribonucleic Acid
GAN	Generative Adversarial Network
GISAID	Global Initiative on Sharing All Influenza Data
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
ORF	Open reading frames
ReLU	Rectified Linear Unit
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
SARS-CoV-2	severe acute respiratory syndrome coronavirus 2
UTR	Untranslated region

0 Project Setup

For a detailed description of how to set up the project, please have a look at `https://github.com/nilskre/bomberman_rl/blob/master/README.md`.

1 Introduction

2 Fundamentals and Related Work

2.1 Basics from molecular biology

This chapter gives a brief introduction into the underlying biological domain.

2.1.1 Genome sequences

The human genome is encoded as Deoxyribonucleic Acid (DNA). The DNA carries genetic instructions how DNA based organisms develop and behave. It is structured as a double helix and consists of two nucleotide pair combinations: adenine (A), thymine (T) and cytosine (C), guanine (G). [12, p. 8]

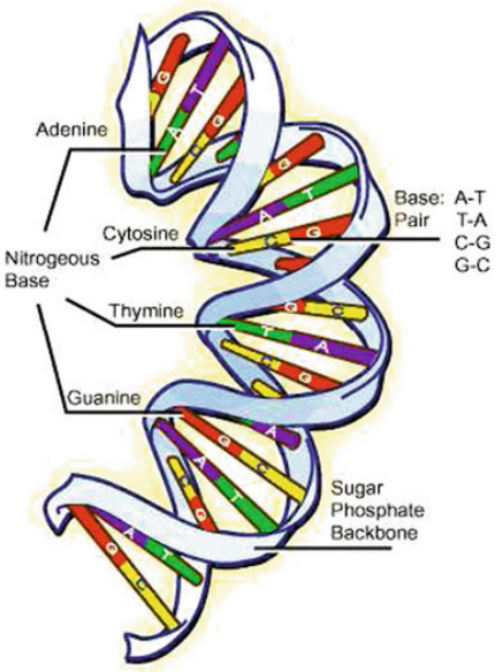

Figure 1: DNA double helix [12, p. 8]

2.1.2 Protein biosynthesis

The protein biosynthesis (also known as the central dogma of molecular biology) describes how proteins are produced based on DNA sequences. The proteins then produce characteristics (e.g. human hair color). So the DNA directly influences which human characteristics are developed. [19, p. 6]

The process of the protein biosynthesis can be seen in figure 2. The single steps are described in the following:

1. Transcription: The DNA is transcribed by the RNA-Polymerase into pre-messenger RNA. As part of this process the nucleotide thymine is replaced by uracil (U). [19, p. 9]

2. mRNA Processing: The pre-messenger RNA is transformed into the messenger RNA by removing the non-coding regions (introns). So the main difference between DNA and Ribonucleic Acid (RNA) is that the DNA is structured as a double helix, whereas the RNA consists of a single strain. [19, p. 9]

3. Translation: Ribosomes translate the mRNA into amino acids. Three nucleotides (one codon) decode one amino acid. The matching which codons decode for which amino acid can be seen in figure 3. [19, p. 9]

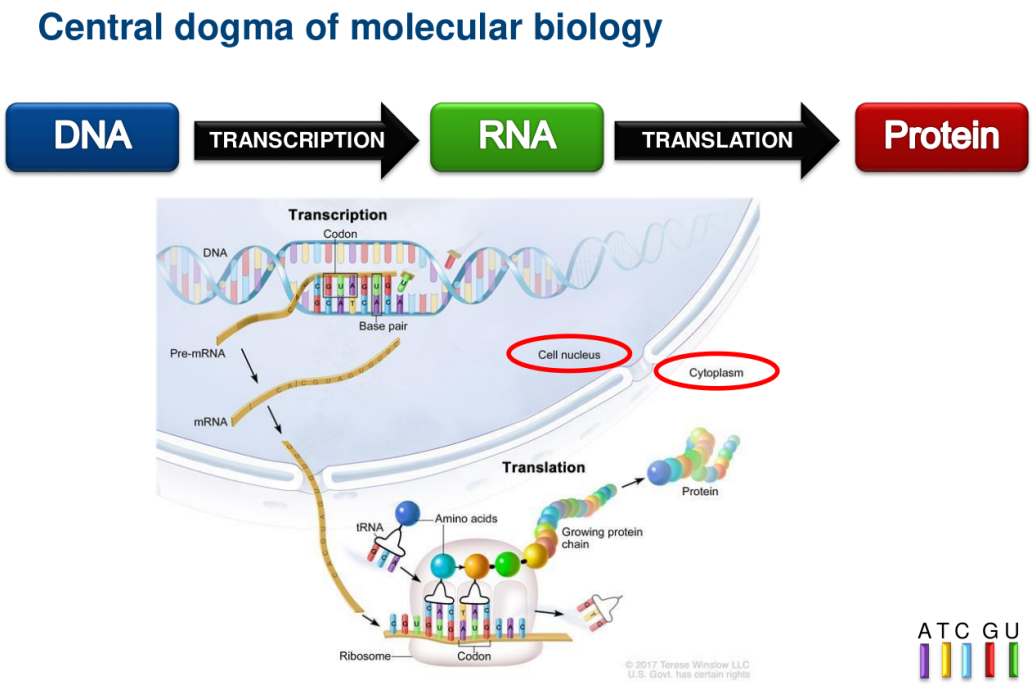

Figure 2: Protein biosynthesis [19]

2.1.3 Structure of SARS-CoV-2

RNA based viruses (like SARS-CoV-2) use RNA to encode their genetic information. That means that the genetic information is present as a single strained RNA sequence. This sequence contains for example the information

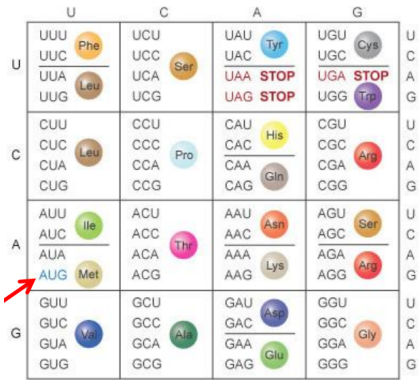

Figure 3: Genetic code [19]

which proteins form the surface structure of the SARS-CoV-2 virus and how this surface looks like. [16]

Based on Naqvi et al. [16] the four proteins forming SARS-CoV-2 are:

- spike (S)
- envelope (E)
- membrane (M)
- nucleocapsid (N)

A structural representation of SARS-CoV-2 and a host cell is visualized in figure 4. [16]

Furthermore figure 4 shows the RNA in the center. The SARS-CoV-2 RNA consists of about 30,000 nucleotides, which belong to different subgroups, as shown in figure 5. Padded by a 5' Untranslated region (UTR) at the beginning and a 3' UTR in the end, the middle part contains 12 functional Open reading frames (ORF), which contain the four coding regions for the proteins S, E, M and N.

2.1.4 Relationships between biological objects

One main question in biology is to determine relationships between objects, such as individuals or genome sequences. The observed objects are called taxa. Most commonly used for determining these relationships is the phylogenetic analysis which generates a phylogenetic tree. In this phylogenetic tree each leaf node corresponds to exactly one taxa. The inner nodes of the tree are the infered hypothetical ancestors, which are no taxas. The relatedness between different taxas can be evaluated by their distance. [6]

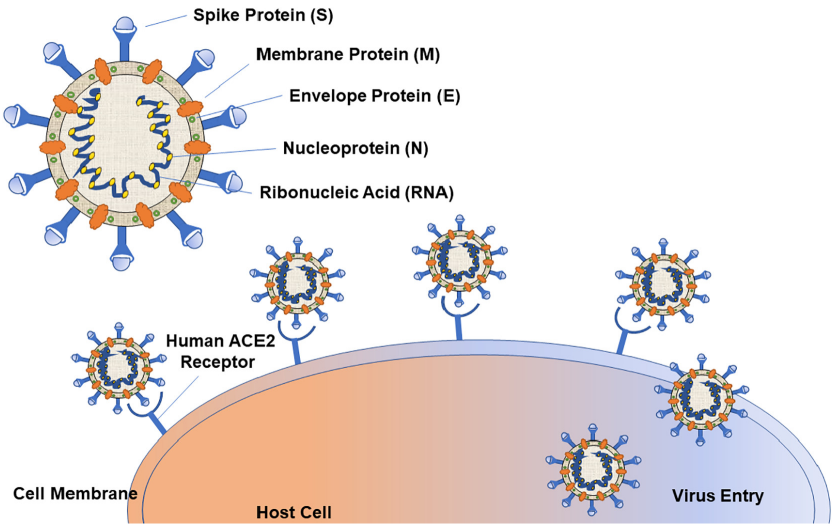

Figure 4: SARS-CoV-2 structure [16]

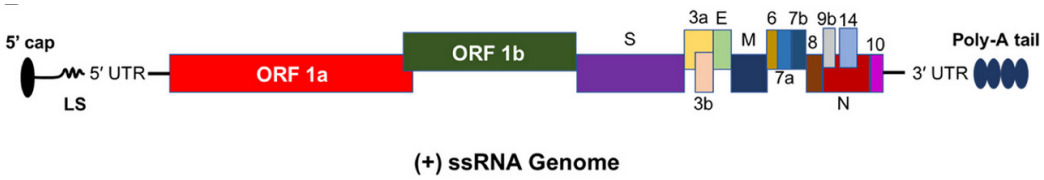

Figure 5: SARS-CoV-2 genome structure [16]

Figure 6 shows an example phylogenetic tree calculated based on the species genomes. On the right side each leaf node (taxa) corresponds to one current species. From left to right one can see the infered historical development from the ancestors to the current species. One can see, that Gorillas and Orangutans have earlier developed apart then the other species. Furthermore one can see, that the Chimpanzees and Bonobos share a most recent common ancestor (inner node next to both). That is why they are more related to each other. [13]

2.2 From Probabilistic Language Models to modeling Evolution Theory

A probabilistic language model tries to approximate the probability distribution

$$P(w_1, ..., w_n) = \Pi_{t=1}^{n} P(w_t | w_1, ..., w_{t-1}) \tag{1}$$

with w_t being a word at position (time step) t in a sentence of length n. To build language models Recurrent Neural Networks (RNNs) were used to

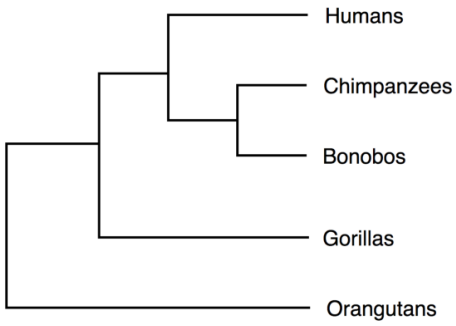

Figure 6: Example phylogenetic tree [13]

model such probability distributions.

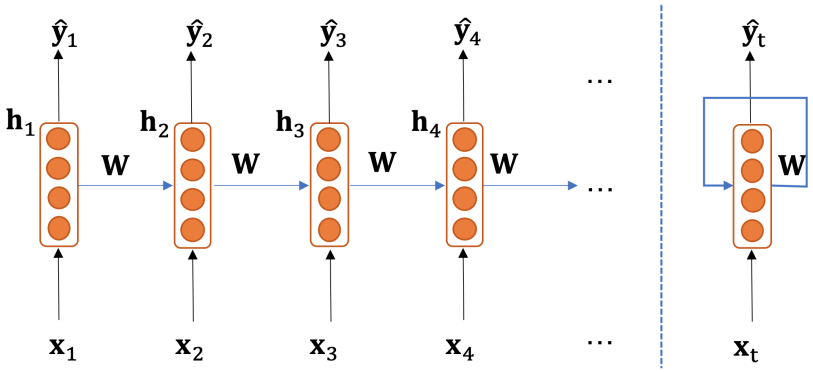

Figure 7: Architecture of a conventional RNN [8]

At each time step t it outputs a probability distribution $P(w_t|w_1, ..., w_{t-1})$ given the words read so far in the current instance (see Figure 7). Words are read as a vectorized numerical representation, often given by pretrained so-called word embeddings x_t which are lower dimensional and more semantically-enriched compared to simple one-hot encodings. One then calculates the hidden state h_t by

$$h_t = f(W^{(h)}h_{t-1} + W^{(x)}x_t + b_1) \tag{2}$$

and the corresponding output probability distribution by

$$\hat{y}_t = softmax(U^{(h)}h_t + b_2). \tag{3}$$

The applied weight matrix is always the same for each time step t giving the RNN its name. One can therefore simplify the unrolled RNN architecture on the left side of Figure 7 to the one on the right, where the hidden state is continuously passed as an input to the next time step. To achieve a better convergence behavior during training, one can also provide the expected hidden state of time step $t-1$ instead of using the predicted hidden state, which is called teacher forcing. RNNs are able to process input of arbitrary length and are by their recurrent character capable to use information from previous time steps. Unfortunately, they are vulerable to vanishing and exploding gradient problems. Long Short-Term Memory (LSTM) is a special RNN architecture that solves such vulnerabilities by owning a separate long-term cell state besides a short-term hidden state and is introduces in subsection 2.6. It is able to preserve information over many time steps. [8]

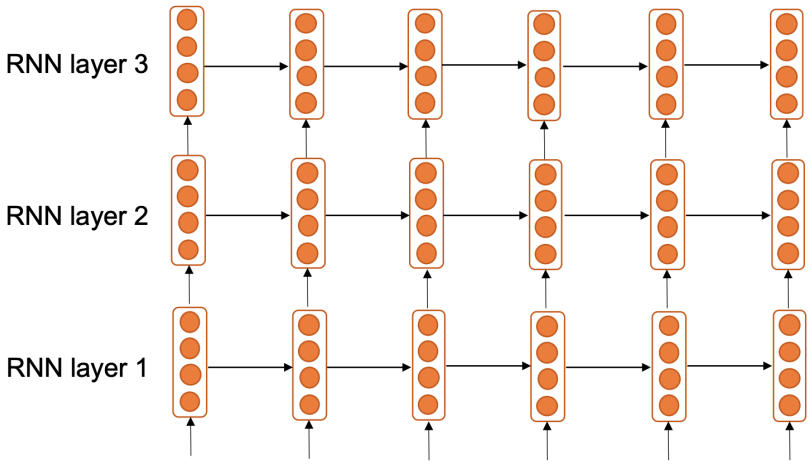

Figure 8: Architecture of a multi-layer RNN [8]

One can also use two RNNs, one traversing a sentence from left to right and another one vice versa, with two different weight matrices to model the probability distribution bidirectionally. One therefore simply concatenates the hidden states of each RNN before applying the the weight matrix U and the $softmax()$ function. Also multi-layer RNN can be utilized to generate higher-order features (hidden states) for the prediction task (see Figure 8). [8]

The RNNs or even better LSTMs architectures used for probabilistic language modeling can be reused in a more complex domain called sequence to sequence modeling for neural machine translation from one language to another. Here one first tries to learn a fixed-dimensional input representation

from an input sequence using an encoder architecture based on an LSTM. The so-called context vector is then decoded by a second LSTM into a new sequence of words preserving the grammar but owning a different meaning. [22]

Sequence to sequence models are introduced together with the LSTM architecture in subsection 2.6. Here the connection to evolution theory can be drawn. RNA sequences made of a concatenation of nucleotides [1] can be represented textually using the FASTA format. A sequence to sequence model can then transferably be applied in the domain of RNA sequences to model how RNA-based viruses change their structure to avoid the detection by the human immune system but still to preserve their infectivity and evolutionary fitness [10].

2.3 GISAID EpiFlu Data Platform

The data needed for this project consists of the SARS-CoV-2 genome data and the corresponding metadata. The most complete database is hosted by the Global Initiative on Sharing All Influenza Data (GISAID) initiative. [2]

The main aim of GISAID is to enable the fast sharing of epidemic and pandemic virus data. In comparison to other databases such as GenBank the genome sequences are shared fast through GISAID. For enabling this fast sharing the GISAID database is only usable after registration, whereas the GenBank database is publicly available. [20]

This leads to the fact, that until september over three million SARS-CoV-2 genome sequences are available through GISAID, whereas only about one million SARS-CoV-2 genome sequences are available through GenBank. Figure 9 shows the selection view of the GISAID database. [14, 1]

2.4 Domain-Specific Methodologies to create Evolutionary Datasets for Mutation Prediction

From databases such as GISAID or GenBank an unordered list of genome sequences and metadata can be downloaded. For training a Machine Learning (ML) model to detect changes and predict future mutations the dataset must contain parent child pairs.

Berman et al. [5] have done this based on two steps. First they created a phylogenetic tree from the genome sequences. Therefore first all genome

[1] We restrict the representation of nucleotides solely to their nucleobases parts consisting of the distinct nucleobases guanine, adenine, cytosine and thymine. We therefore do not include the phosphate group and the five-carbon sugar components.

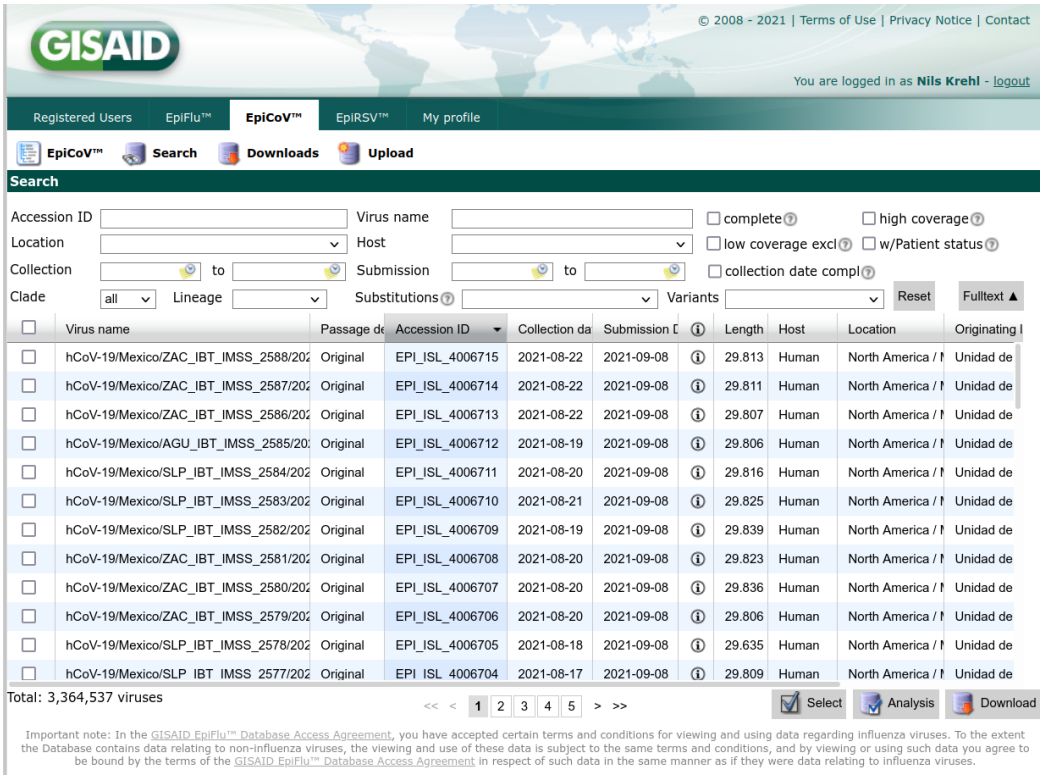

Figure 9: GISAID database

sequences are aligned. The initial phylogenetic tree is calculated by Fasttree using the approximate maximum likelihood method. In the next step the tree is refined by a generalized time-reversible (GTR) model and a gamma model. The final phylogenetic tree is achieved by optimizing the branch lenght and the used models for tree generation. Like in standard phylogenetic trees each leaf node corresponds to one data instance, whereas the inner nodes are infered from the other data instances. Secondly based on the rooted phylogenetic tree, parent-child pairs are calculated. Therefore first a marginal ancestral sequence reconstruction is executed based on the phylogenetic tree from the previous step. Now also the inner nodes correspond to data instances. In the next step BioPython's [7] Bio.Phylo package is used to determine parent-child pairs. From each edge one parent-child pair is generated. [5]

Mohamed et al. [15] used existing datasets from Ogali et al. [17] in their work about mutation prediction. Ogali et al. [17] performed a phylogenetic analysis. After selecting only genome sequences with high quality, they removed duplicate sequences and added reference sequences to the dataset. Now the alignment is executed. In the next step MEGA (Molecular Evolutionary

Genetics Analysis) is used to create the phylogenetic tree. This is done by using the maximum likelihood method, the best-fit general time-reversible (GTR) model and gamma-distributed rate variation.

Hadfield et al. [9] have created the nextstrain project. It aims to examine pathogen's spread and evolution by providing a viral genome database, a bioinformatics pipeline for phylodynamics analysis and a visualization platform. Steps executed by the bioinformatics pipeline for phylodynamic analysis are: "subsampling, alignment, phylogenetic inference, temporal dating of ancestral nodes and discrete trait geographic reconstruction, including inference of the most likely transmission events" [9]. For calculating the phylogenetic tree, TreeTime's maximum likelihood method is used. [9]

2.5 Previous Work on Mutation Prediction

Even before the rise of Covid-19 there had been studies trying to predict mutations of RNA viruses. In the collection of [25, 24, 26] the authors predict the mutation positions in hemagglutinins from influenza A virus using logistic regression and plain neural networks and then use the resulting amino acid mutating probabilities to derive possible mutated amnio acids. The same approach is further used for H5N1 neuraminidase proteins.

Salama et al. [18] proved that nucleotides in an RNA sequence can change based on their local neighborhood. Neural networks are used to predict new strains of the Newcastle virus and subsequently a rough set theory based algorithm is introduced to extract the according point mutation patterns.

Mohamed et al. [15] used a more modern sequence to sequence approach based on LSTMs to learn nucleotide mutations between time-series species of H1N1 Influenza virus and the Newcastle virus as mutations can also be influenced by long-distance relations of amino acids. Therefore one hot-encoded RNA sequences of a parent generation preprocessed to words is given as an input and the output is the predicted offspring generation evaluated by accuracy to the compared true offspring generation. The achieved accuracy in this paper is questionably high with 98.9% on the H1N1 Influenza virus and 96.9% on the Newcastle virus, possibly because of overfitting to the few 4.609 samples for H1N1 Influenza virus and only 83 for the Newcastle virus. Our approach therefore tries to increase the number of samples available for training when building the dataset.

Our approach will neither use any of the just mentioned architectures, but uses a transformer based architecture coupled with a GAN-style training architecture. Nevertheless a short introduction into sequence to sequence models and the underlying long short-term memory components shall be given to better point out our architectural decisions .

2.6 Sequence to Sequence Models based on Long Short-Term Memory

The original LSTM unit was introduced in [11] and can be used for language modeling instead of using plain RNNs to prevent running into vanishing or exploding gradient problems [21]. The architecture of an LSTM is shown in the following figure:

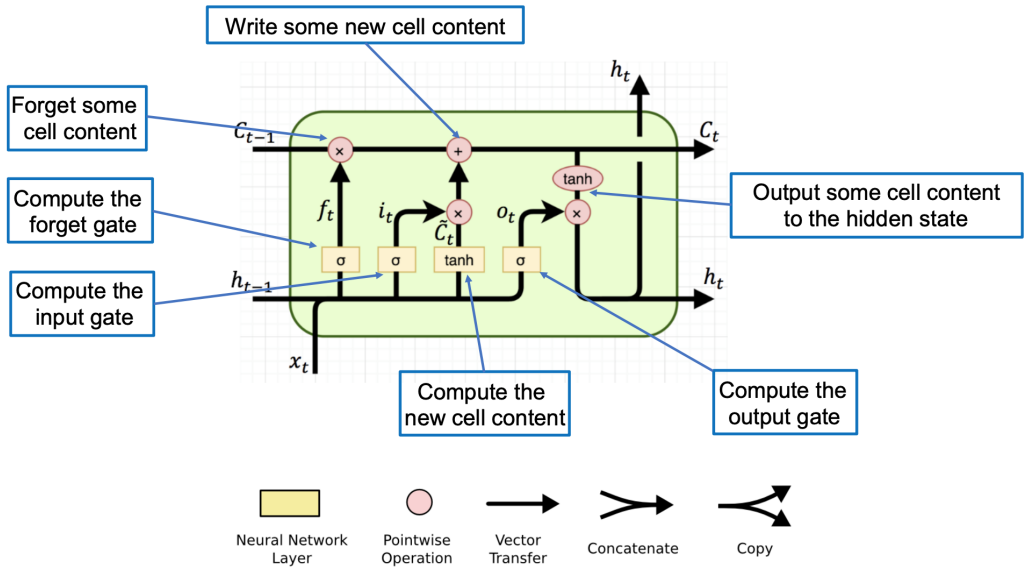

Figure 10: Architecture of an LSTM [8]

It consists of a hidden state h_t and an additional cell state c_t. The cell state stores long-term information and is used to derive a new hidden state. Information flows through three different gates inside the LSTM. The forget gate is used to control which parts of the cell state are potentially carried on to the next time step, the input gate is responsible to decide which parts of the cell state should be updated and the output gate determines what is being passed on as the new hidden state. All three gates and depend on the previous hidden state and the current input. They provide factors limited to the interval [0,1] by the sigmoid function and are multiplied with the cell state, the changes to be added to the cell state and the new hidden state derived from the cell state. Through the cell state an LSTM therefore makes it possible to capture long-distance dependencies. [8]

[22] introduced sequence to sequence learning following a multi-layer encoder-decoder style model architecture. One layer consists of one LSTM that is used as an encoder to learn a large fixed-dimensional vector representation of a size-unrestricted input sequence called the context vector. This vector

consists of the last cell and hidden state of the encoder and incorporates the structure of the input sequence helping the followig decoder LSTM to provide qualitative predictions for the output sequence. The second LSTM therefore serves as a beam search[2] decoder to map the context vector to a corresponding output sequence whose length does not need to match with the length of the input sequence. The output probabiity distribution is therefore given by the equation

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \Pi_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1}) \tag{4}$$

with v being the context vector. Using an LSTM is prefered over a normal RNN as it is used to capture the long range temporal dependencies of the input data. The encoder-decoder architecture uses four layers in total partitioned onto four Graphics Processing Units (GPUs). A corpus of 160k words for the input sequence and another one of 80k words for the target sequence was used to create the word embeddings of dimension 1000. Unknown words were replaced by a *UNK* token. The sequence to sequence model approach was evaluated for neural machine translation and reached a 34.81 BLEU score. One finding during training was that reversing the input sequence introduces many short term dependencies as the minimal time lag of the problem is reduced making optimization easier. [22]

2.7 Applying Generative Adversarial Networks

Using a plain sequence to sequence model for mutation prediction does not necessarily guarantee that the generated sequences are evolutionary offsprings of a parent generation as not being included as is in the ground truth data. The generated sequences might occur realistic and biologically relevant, but a plain sequence to sequence architecture does not inherently check for natural parental descent and therefore does not make sure whether the predicted mutations lead to improved fitness. [5] developed a novel sequence to sequence framework based on the Generative Adversarial Network (GAN) idea to predict genetic mutations and future biological polulations of the influenza virus (see Figure 11). MutaGAN describes a sequence to sequence generator within an adversarial framework that predicts protein sequences augmented with possible mutations. By using a sequence to sequence generator and a discriminator specialized on separating fake evolutionary mutations from real ones, one can then guarantee to a certain degree that the evolutionary

[2]Do not choose the most probable word but the B most likely word hypothesis and pass them to the next time step in the LSTM. Whichever hypothesis results in the lowest loss is kept. To avoid combinatorial explosion limit the beam depth size.

parent-childhood coherence is given. In MutaGAN a mutation is considered correct if the change in amino acid and location within the RNA sequence is equal to the parent's true offspring. [5]

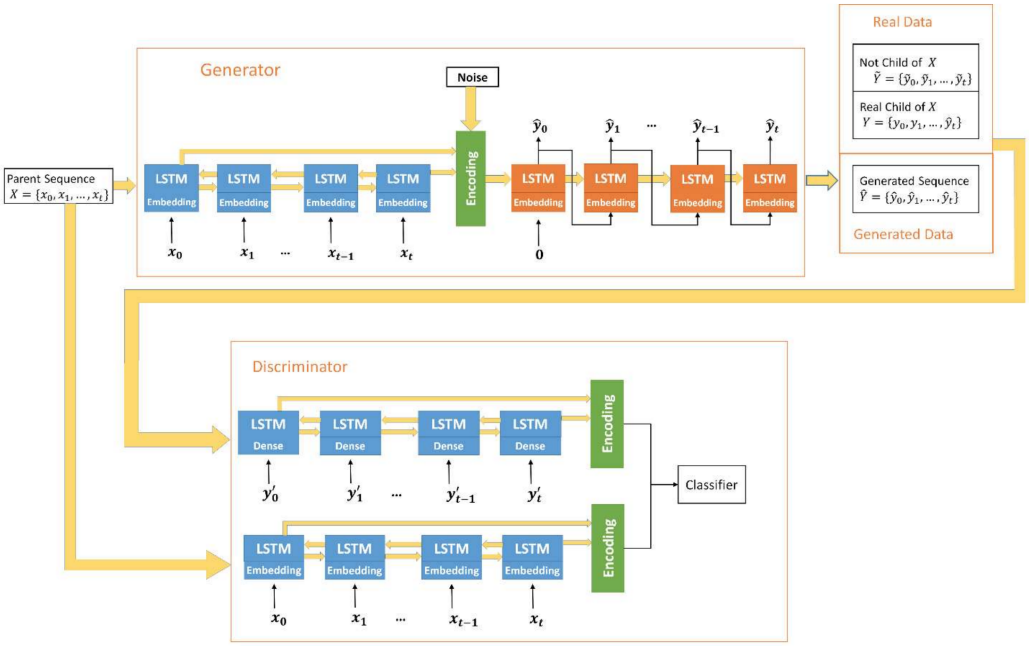

Figure 11: Architecture of MutaGAN [5]

MutaGAN's generator consists of a sequence to sequence model. The encoder is built from of a bidirectinal LSTM and the resulting context vector of dimensionality 512 is combined with noise from a standard normal distribution. The decoder LSTM predicts the resulting sequence of amino acids greedily using the $argmax()$ of the $softmax()$ output for every time step. The discriminator is trained on three different parent-child pair configurations to optimally compete against the generator, real parent-child pairs, real parent and generated child pairs and pairs of real sequences that are not parent-child pairs. Two bidirectional LSTM encoders sharing the same weight matrix as the encoder of the generator produce a fixed-dimensional encoding of the provided parent-child pair used for classification of evolutionary descent by a plain neural network. The child encoder in the discriminator uses a dense layer having the same weight matrix as the embedding layer of the encoder of the generator. The dense layer is required to directly input the predicted child sequence as its probability distribution rather than the final output after applying the $argmax()$ as it would not enable backpropagation to train the generator. In case of a true child sequence is given a simple one-hot

encoding is used. [5]

Interestingly [5] states planned improvements by utilizing bigger datasets aquired from the GISAID database EpiFlu and by using more length-robust attention-based models to directly work on nucleotide sequences instead of sequences of amino acids. Therefore transformers and the attention mechanism are introduced in the following section.

2.8 Transformer and Attention Mechanism

Using a sequence to sequence model as introduced based on an encoder-decoder architecture of LSTM cells has some drawbacks. First the input needs to be process sequentially in time steps which makes parallelization difficult and increases training time, especially for longer sequences. Furthermore the hidden and cell state vector passed through every timestep tries to encode information of *all* previous time steps without knowing which information is especially important for the current time step. This not only makes long distance dependencies hard to capture, but also might not get the prioritization of the previous time step inputs right. [4, 23]

To tackle these problems the so-called transformer architecture was introduced in [23]. It feeds an entire sequence into the encoder to be processed in parallel denying any concept of recurrence or convolution. Only using a so called self-attention mechanism the transformer makes sure that during processing every input position, each of them receives the information that is most important to them. This way modeling long dependencies becomes much more easy compared to LSTMs as the view on the input sequence is more global. In this architecture, the encoder also passes all computed hidden states of every position to the decoder, which therefore [23] can generate the target sequences based on more semantically enriched features and also in parallel for every position. [23]

The transformer architecture achieved state-of-the-art quality results with a BLEU score of 41.8 on the WMT 2014 English-to-French translation task (cf. BLEU score of [22] was 34.8), while still being much faster to train due to its parallelization capabilities. State-of-the-art results are already achieved after just twelve hours of training on eight P100 GPUs. As this is still far beyond the scope and resources given for this project, this projects provides a proof-of-concept transformer model trained for far shorter and fewer sequences as one would need to predict entirely new RNA sequences. [23]

First the transformer architecture should be introduced in the following:

The transformer consists of an encoder and a decoder part just as normal sequence to sequence models. One layer of the encoder contains a self-

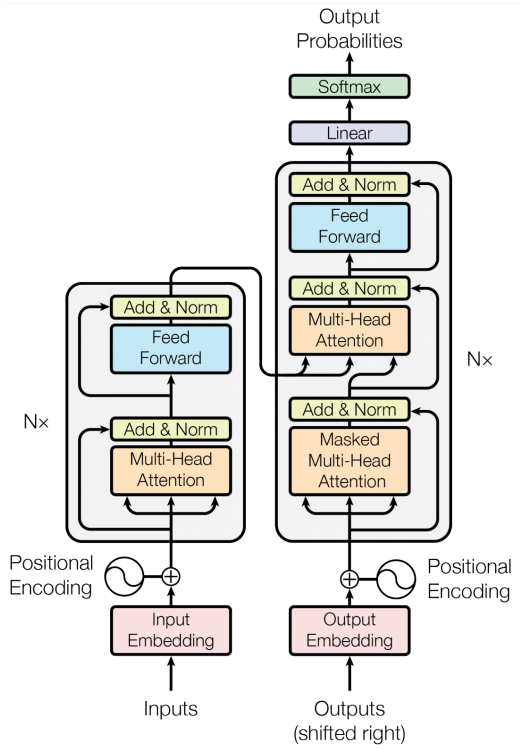

Figure 12: Architecture of the transformer [23]

attention layer before passing the hidden state representations of an input sequence to a feed forward neural network consisting of two linear transformations separated by Rectified Linear Unit (ReLU). Thus it makes sure that semantic and contextual dependencies between different positions in the input sequence are also modeled. Also skip connections are added for both components with subsequent layer normalization. [3, 23]

To calculate the self-attention ouput of a sequence, a query Q, key K and value V matrix is calculated through multiplication of the sequence representation[3] with learned transformation matrices. Each row of the three received matrices corresponds to a specific input position. The output of the self-attention layer is then calculated through

$$(ScaledDot - Product)Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \quad (5)$$

The dot product between Q and K calculates scores that are used as weighting factors for V. This models for every input position how relevant

[3] A matrix containing the hidden state representations of every input position

all other input positions are for each of the hidden state encodings. The scores are divided by the square root of the dimensionality of the query/key values of the hidden state representations, which is chosen to be 64 (square root eight), to receive more stable non-vanishing gradients. The softmax guarantees that the positional scores lie between zero and one and add up to one. Note that the score of the current input position itself will most likely have the hightest score. [3, 23]

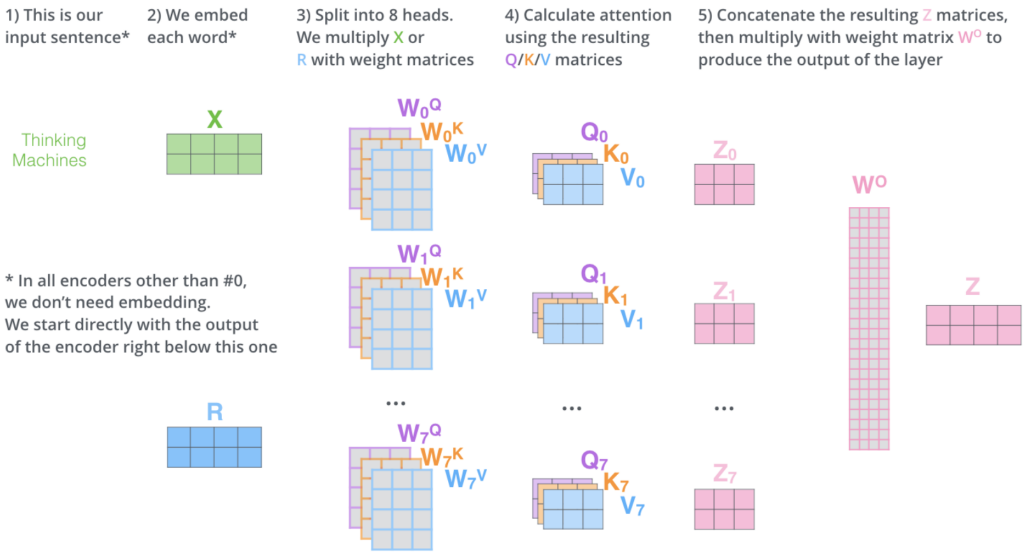

Figure 13: Multi-head attention architecture [3]

One can expand self-attention towards a multi-head attention. In this configuration each head produces its own query, key and value matrices and therefore its own self-attention output in its own representation subspace. The resultung hidden state is calculated by concatenating each hidden state output of the different heads and multiplying it with a contraction matrix that outputs a hidden state vector in its single-head dimensionality. Multi-head attention is needed to be able to better focus on multiple parts of the input sequence when encoding an input position, otherwise it might happen that most of the focus is set on the input position itself. [23] uses 8 heads. [3, 23]

Overall six layers of encoder and decoder components having individual weight matrices are stacked on top of each other to capture low-level as well as high-level features. The dimensionality of the hidden state is 512. Note that each position of the input sequence is encoded on an individual path throught the transformer, which makes parallelization possible compared to LSTM based architectures. [3, 23]

The decoder uses almost the same architecture as the encoder, but contains an additional multi-head attention component that integrates all the hidden state encodings from the encoder. Therefore the final hidden state encodings resulting from the top most encoder are transformed into key and value attention vectors and given to the integrating mutli-head attention component of every decoder. Once again this helps to better focus on the relevant input positions for the current output position based on hidden state information extracted from the encoder. Note that the decoder works in a sequential manner again meaning that the input to the decoder are the already decoded sequence tokens of previous time steps, future input positions are masked away. The first multi-head attention component therefore capture inter-dependencies in the generated output sequence, the second one, the so called encoder-decoder attention, enriches the hidden state encoding with the information given from the encoder hidden states. Finally a linear layer mapping the hidden state vector to a logits vector with dimensionality of the given output dictionary and a $softmax()$ function produce the output sequence using the greedy or beam search approach. The EOS token symbolizes the end of the output to be generated. [3, 23]

Input to the transformer are word embeddings also of fixed dimensionality of 512. Onto each word embedding a positional encoding following a specific pattern is added. These patterns are either learned or fixed and make sure that the distances of the input positions are projected onto the queue, key and value representations to be utilized during scaled dot-product attention. This also enables self-attention to be scaled to unseen lengths of sequences. Transformers usually contain an upper bound of sequence length to be processed, which is different to recurrent architectures that can process inputs of arbitrary length. Input sequences are usually padded up until the specified sequence length, the used PAD tokens are masked to be excluded from the self-attention components. [3, 23]

2.9 Other Techniques

- NNs/SVMs: `https://bsb-eurasipjournals.springeropen.com/articles/10.1186/s13637-016-0042-0`
- BiLSTM: `https://science.sciencemag.org/content/371/6526/284`

3 Approach

TODO: Pipeline image

3.1 Dataset Creation

hier irgendwo das Zielschema des Datensatzes beschreiben

3.1.1 Raw data selection from GISAID

focus: Germany from 4.5. - 6.8. (new variant arised in the recent past -> lambda, delta, ...) only Germany, to make it possible to handle the data about 35000 genomes in our raw dataset

beispiel record: genome sequence and metadata

3.1.2 Generation of a phylogenetic tree

3.1.3 Phylogenetic tree to dataset

3.2 Data Preprocessing

two steps: - to make the dimensionality managable not the whole 30000 nucleotides are evaluated. We take a subpart of X nucleotides from position A to B - Transform string to numeric for model input

3.2.1 Dimensionality reduction by selecting subpart of the genome

3.2.2 Transform genome sequence to numeric model input

- DNA Sequencing (Done during dataset creation, given from GISAID)
- DNA Sequence Tokenization for Amino Acid Dictionary
- DNA Sequence Padding

3.3 Model Architecture

3.4 Training Process

4 Experimental results

5 Conclusion

References

[1] GISAID (editor). *GISAID*. Sept. 2021. URL: https://www.gisaid.org/ (visited on 09/08/2021).

[2] GISAID (editor). *GISAID - Mission*. GISAID mission. 2021. URL: https://www.gisaid.org/about-us/mission/ (visited on 07/05/2021).

[3] Jay Alammar. *The Illustrated Transformer*. 2018. URL: https://jalammar.github.io/illustrated-transformer/ (visited on 08/18/2021).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv* (2016). URL: https://arxiv.org/abs/1409.0473 (visited on 08/19/2021).

[5] Daniel S. Berman et al. "MutaGAN: A Seq2seq GAN Framework to Predict Mutations of Evolving Protein Populations". In: *arXiv* (2020). URL: https://arxiv.org/abs/2008.11790 (visited on 07/05/2021).

[6] H.J. Böckenhauer and D. Bongartz. *Algorithmische Grundlagen Der Bioinformatik: Modelle, Methoden Und Komplexität*. XLeitfäden Der Informatik. Vieweg+Teubner Verlag, 2013. ISBN: 978-3-322-80043-5. URL: https://books.google.de/books?id=2HL3BQAAQBAJ.

[7] Peter J. A. Cock et al. "Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics". In: *Bioinformatics* 25.11 (Mar. 2009), pp. 1422–1423. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp163. eprint: https://academic.oup.com/bioinformatics/article-pdf/25/11/1422/944180/btp163.pdf. URL: https://doi.org/10.1093/bioinformatics/btp163.

[8] Michael Gertz. "Text Analytics - Text Classification". University Heidelberg, 2020.

[9] James Hadfield et al. "Nextstrain: Real-Time Tracking of Pathogen Evolution". In: *Bioinformatics* 34.23 (May 2018), pp. 4121–4123. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty407. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/23/4121/26676762/bty407.pdf. URL: https://doi.org/10.1093/bioinformatics/bty407.

[10] Brian Hie et al. "Learning the Language of Viral Evolution and Escape". In: *Science* 371.6526 (2021), pp. 284–288. ISSN: 0036-8075. DOI: 10.1126/science.abd7331. URL: https://science.sciencemag.org/content/371/6526/284 (visited on 07/05/2021).

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *ResearchGate* (1997). URL: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory (visited on 08/13/2021).

[12] Nan M. Laird and Christoph Lange. *The Fundamentals of Modern Statistical Genetics*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 978-1-4419-7337-5.

[13] Vijini Mallawaarachchi. *Molecular Phylogenetics Using Bio.Phylo*. Medium. Mar. 24, 2018. URL: https://towardsdatascience.com/molecular-phylogenetics-using-bio-phylo-57ce27492ee9 (visited on 09/09/2021).

[14] National Library of Medicine (NCBI) (editor). *NCBI SARS-CoV-2 Resources*. URL: https://www.ncbi.nlm.nih.gov/sars-cov-2/ (visited on 09/08/2021).

[15] Takwa Mohamed et al. "Long Short-Term Memory Neural Networks for RNA Viruses Mutations Prediction". In: *Hindawi* (2021). URL: https://www.hindawi.com/journals/mpe/2021/9980347/ (visited on 08/11/2021).

[16] Ahmad Abu Turab Naqvi et al. "Insights into SARS-CoV-2 Genome, Structure, Evolution, Pathogenesis and Therapies: Structural Genomics Approach". In: *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease* 1866.10 (2020), p. 165878. ISSN: 0925-4439. DOI: 10.1016/j.bbadis.2020.165878. URL: https://www.sciencedirect.com/science/article/pii/S092544392030226X.

[17] Irene N. Ogali et al. "Molecular Characterization of Newcastle Disease Virus from Backyard Poultry Farms and Live Bird Markets in Kenya". In: *International Journal of Microbiology* 2018 (Aug. 5, 2018). Ed. by Simona Nardoni, p. 2368597. ISSN: 1687-918X. DOI: 10.1155/2018/2368597. URL: https://doi.org/10.1155/2018/2368597.

[18] Mostafa Salama, Aboul Ella Hassanien, and Ahmad Mostafa. "The Prediction of Virus Mutation Using Neural Networks and Rough Set Techniques". In: *EURASIP Journal on Bioinformatics and Systems Biology* 2016 (2016). DOI: 10.1186/s13637-016-0042-0.

[19] Dominique Scherer and Justo Lorenzo Bermejo. "Statistical Genetics and Genetic Epidemiology - SNPs and Other Common Variants". lecture manuscript. lecture manuscript. University Heidelberg, 2021.

[20] Yuelong Shu and John McCauley. "GISAID: Global Initiative on Sharing All Influenza Data – from Vision to Reality". In: *Eurosurveillance* 22.13 (Mar. 30, 2017). ISSN: 1560-7917. DOI: 10.2807/1560-7917.ES.2017.22.13.30494. URL: https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2017.22.13.30494 (visited on 09/08/2021).

[21] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM Neural Networks for Language Modeling". In: *ISCA Archive* (2012). URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2012/i12_0194.pdf (visited on 08/12/2021).

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *arXiv* (2014). URL: https://arxiv.org/abs/1409.3215 (visited on 05/15/2021).

[23] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv* (2017). URL: https://arxiv.org/abs/1706.03762 (visited on 08/17/2021).

[24] Guang Wu and Shaomin Yan. "Prediction of Mutations Engineered by Randomness in H5N1 Hemagglutinins of Influenza A Virus". In: *SpringerLink* (2007). URL: https://link.springer.com/article/10.1007/s00726-007-0602-4 (visited on 08/13/2021).

[25] Guang Wu and Shaomin Yan. "Prediction of Mutations Engineered by Randomness in H5N1 Neuraminidases from Influenza A Virus". In: *SpringerLink* (2007). URL: https://link.springer.com/article/10.1007/s00726-007-0579-z (visited on 08/13/2021).

[26] Guang Wu and Shaomin Yan. "Prediction of Mutations in H1 Neuraminidases from North America Influenza A Virus Engineered by Internal Randomness". In: (2008). URL: https://link.springer.com/article/10.1007/s11030-008-9067-y (visited on 08/13/2021).