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

List of Abbreviations

GAN	Generative Adversarial Network
GISAID	Global Initiative on Sharing All Influenza Data
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network

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 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 model such probability distributions.

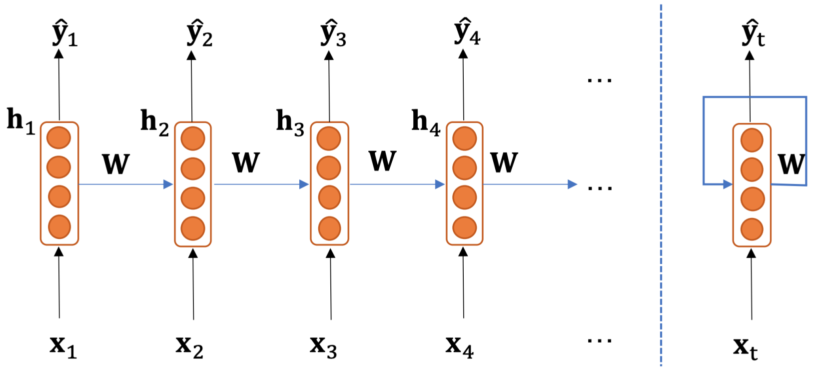

Figure 1: Architecture of a conventional RNN [2]

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 1). 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 porbability 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 1 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 2.5. It is able to preserve information over many time steps. [2]

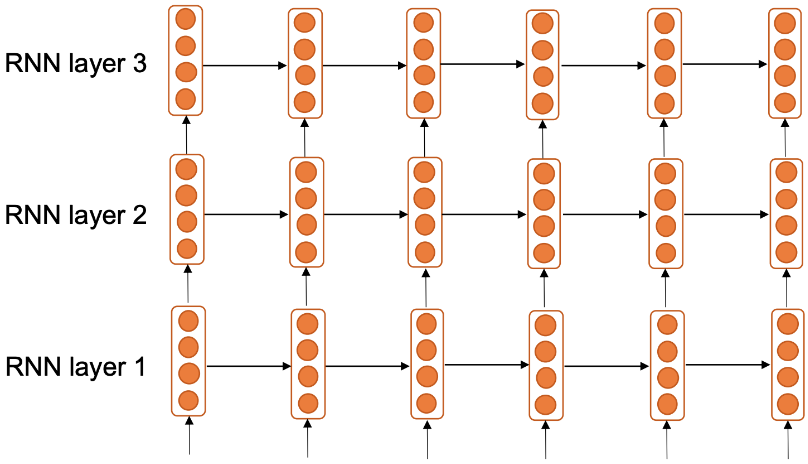

Figure 2: Architecture of a multi-layer RNN [2]

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 2). [2]

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. [8]

Sequence to sequence models are introduced together with the LSTM architecture in subsection 2.5. Here the connection to evolution theory can be drawn. Ribonucleic Acid (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 [3].

2.2 GISAID EpiFlu Data Platform

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

2.4 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 [11, 10, 12] 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.

[6] 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.

[5] uses 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.

[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.

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 .

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

The original LSTM unit was introduced in [4] and can be used for language modeling instead of using plain RNNs to prevent running into vanishing or exploding gradient problems [7]. The architecture of an LSTM is shwon in the following figure:

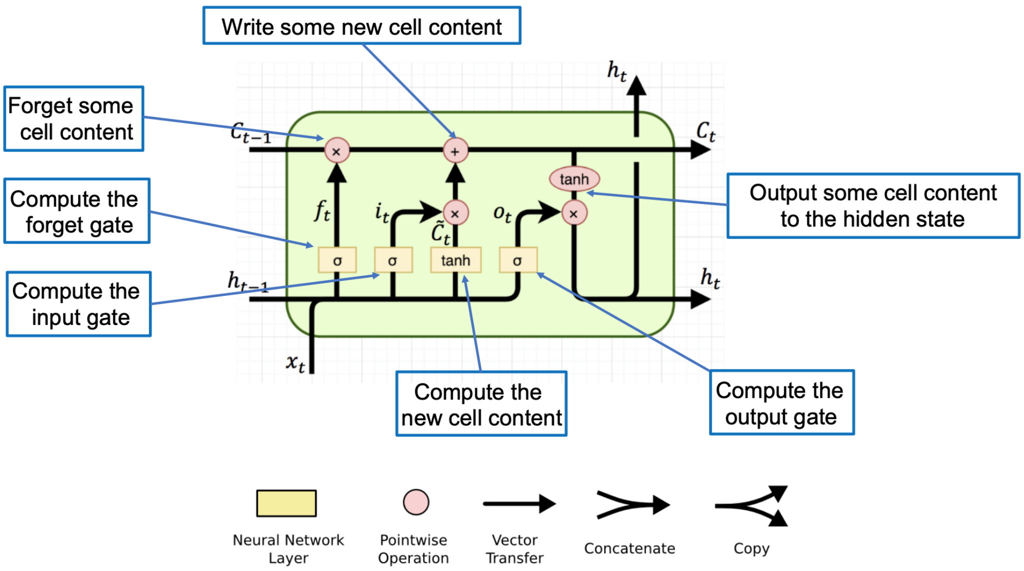

Figure 3: Architecture of an LSTM [2]

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. [2]

[8] 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. [8]

2.6 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. [1] developed a novel sequence

[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. To avoid combinatorial explosion limit the beam depth size.

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 4). 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 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. [1]

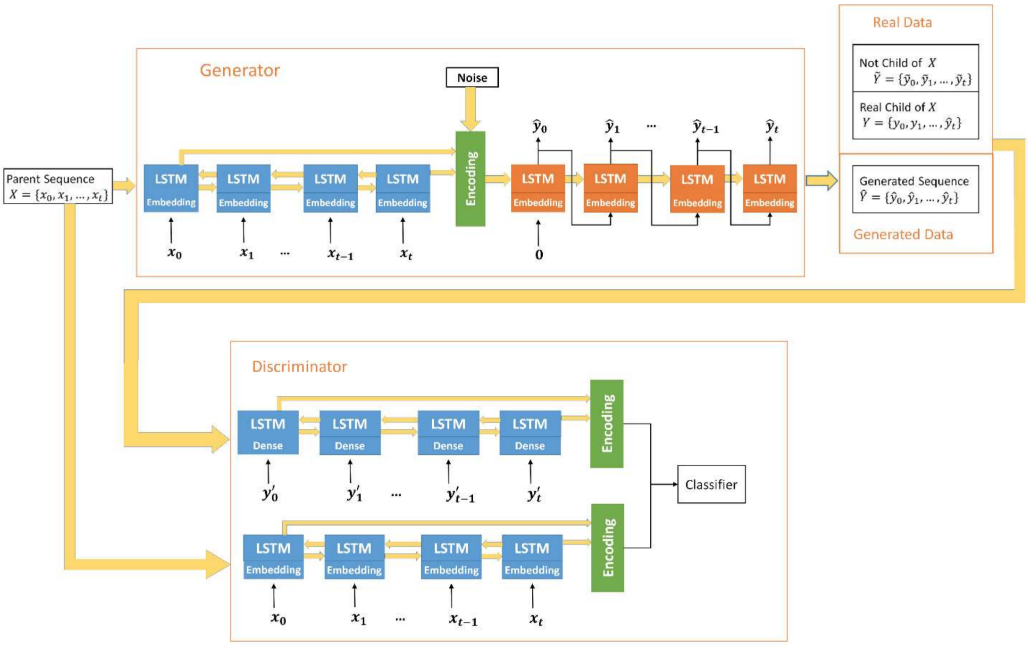

Figure 4: Architecture of MutaGAN [1]

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. [1]

Interestingly [1] states planned improvements by utilizing bigger datasets aquired from the Global Initiative on Sharing All Influenza Data (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.

2.7 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 inportant for the current time step. Also this makes long distance dependencies hard to capture.

To tackle these problems the so-called Transformer architecture was introduced in [9]. 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 can generate the target sequences based on more semantically enriched features and also in parallel for every position. [9]

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 [8] 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 GPU. 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. [9]

First the Transformer architecture should be introduced in the following:

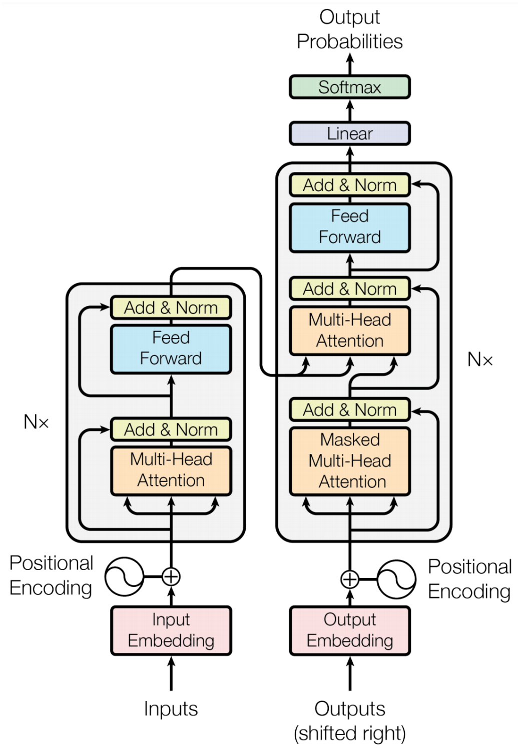

Figure 5: Architecture of the transformer [9]

2.8 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

3.1 Dataset Creation

3.2 Data Preprocessing

- DNA Sequencing
- DNA Sequence Tokenization for Amino Acid Dictionary
- (DNA Sequence Padding not necessary as model can handle input of arbitrary length)
- Phologenetic Tree and final dataset metrics

3.3 Model Architecture

3.4 Training Process

4 Experimental results

5 Conclusion

References

[1] 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).

[2] Michael Gertz. "Text Analytics - Text Classification". University Heidelberg, 2020. (Visited on 08/15/2021).

[3] Brian Hie et al. "Learning the language of viral evolution and escape". In: *Science* (2021). URL: https://science.sciencemag.org/content/371/6526/284 (visited on 07/05/2021).

[4] 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).

[5] 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).

[6] 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). URL: https://bsb-eurasipjournals.springeropen.com/articles/10.1186/s13637-016-0042-0 (visited on 07/05/2021).

[7] 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).

[8] 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).

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

[10] 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).

[11] 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).

[12] 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).