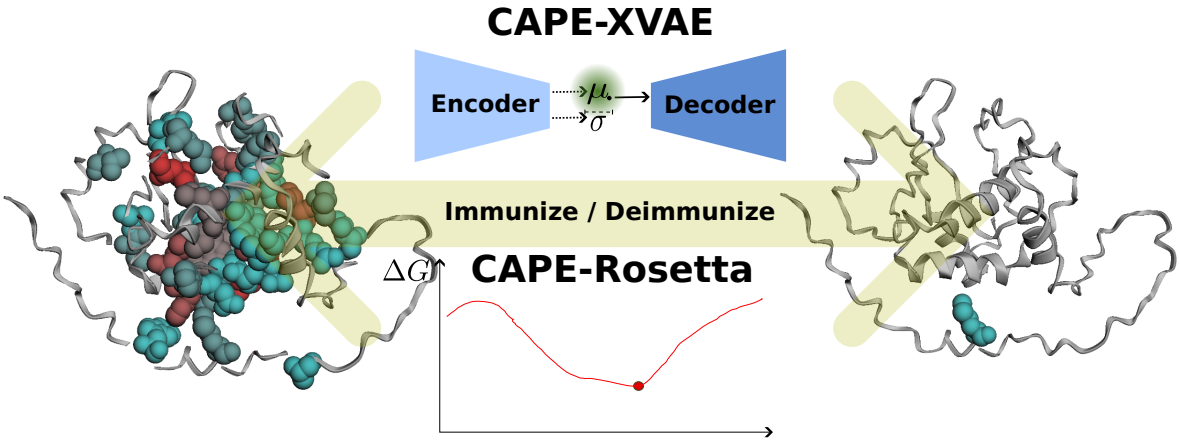
Utility of language model and physics-based approaches in modifying MHC Class-I immune-visibility for the design of vaccines and therapeutics

Hans-Christof Gasser[*,1]
Diego Oyarzun[1,2]
Ajitha Rajan[*,**,1] (co-directing)
Javier Alfaro[*,**,1,3,4] (co-directing)

[1]School of Informatics, University of Edinburgh, Edinburgh, UK, [2]School of Biological Sciences, University of Edinburgh, Edinburgh, UK, [3]International Centre for Cancer Vaccine Science, University of Gdańsk, Gdańsk, Poland, [4]Department of Biochemistry and Microbiology, University of Victoria, Victoria, British Columbia, Canada
[*]Correspondence to: Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB
h.gasser@sms.ed.ac.uk, arajan@ed.ac.uk, javier.alfaro@proteogenomics.ca
[**]co-directing

July 2023

Abstract



Protein therapeutics promise to revolutionize medicine, with an arsenal of applications that include disrupting protein interactions, acting as potent vaccines, and replacing genetically deficient proteins. Therapeutics must avoid triggering unwanted immune responses towards the therapeutic protein or viral vector proteins. In contrast, vaccines require a robust immune reaction targeting a broad range of pathogen variants. Therefore, computational methods modifying proteins' immune visibility, while maintaining functionality, are needed. This paper focuses on visibility to cytotoxic T-lymphocytes, which use the MHC Class I pathway to detect destruction targets.

To explore the limits of modifying MHC-I immune visibility within the distribution of naturally occurring sequences, we developed a novel machine learning technique, `CAPE-XVAE`, that combines a language model with reinforcement learning to modify a protein's immune visibility. Our results show that `CAPE-XVAE` effectively modifies the MHC-I immune visibility of the HIV Nef protein. We contrast `CAPE-XVAE` to `CAPE-Packer`, a physics-based method we also developed, and observe that while the machine learning approach generates more natural appearing sequences, the physics-based approach achieves a greater reduction in immune visibility. Compared to `CAPE-Packer`, `CAPE-XVAE` has a propensity to suggest sequences that draw upon local sequence similarities in the training set. This is particularly important for vaccine development, to be representative of the real viral population and the approach holds promise for preserving both known and unknown functional constraints, which are essential for deimmunization of therapeutic proteins. `CAPE-Packer`, solely focuses on preserving the protein's fold, falling short of capturing this complexity.

Keywords: Deimmunization; Vaccines; MHC Class I; Machine Learning; Language Models

Acronyms

AA amino acid.
AE auto encoder.
AR auto-regressive.
CAPE Controlled Amount of Present Epitopes.
CTL Cytotoxic T-lymphocytes.
MHC major histocompatibility complex.
MHC-I MHC Class I.
MHC-II MHC Class II.
ML machine learning.
PWM position weight matrix.
RL reinforcement learning.
RMSD root mean squared distance.
VAE Variational Autoencoder.

1 Introduction

The recent SARS-CoV-2 pandemic has shown that RNA vaccines are safe, quick to develop and can reliably induce the production of viral proteins by patients. Similar technologies could also be used to stimulate production of other proteins exhibiting a broad range of functions. To illustrate their utility, as of May 4, 2023, the mRNA therapeutics company Moderna is actively developing a range of mRNA vaccines targeting respiratory diseases such as COVID-19, influenza, and RSV, as well as latent infections like CMV, EBV, and Zika. Additionally, they are pioneering mRNA therapeutics for various conditions including autoimmune hepatitis, cancer, and genetic disorders like propionic acidemia and cystic fibrosis [1]. While an immune response to the SARS-CoV-2 spike protein vaccine was a desired reaction, an immune response directed at a protein intended to provide a useful function within target cells could diminish a therapy's effectiveness or, in more severe cases, lead to autoimmune reactions. Therefore, cost-efficient computational methods that are able to adjust the immune-visibility of a target protein, while preserving it's function, will be essential to the success of these therapeutics.

In particular, the exploration of deimmunizing proteins has gained significant attention and the methods employed to reduce immune visibility are by no means limited to bioinformatics techniques. An in depth review covering the whole breadth of methods - also including biological shielding mechanisms - can be found in Zinsli et al. [2]. So far, much of the focus in deimmunization research has been directed towards mitigating immune responses by antibody recognition. This has often been targeted towards antibody epitopes located on the protein's surface. Monoclonal antibodies, for instance, are frequently developed using animal models and, to reduce immune visibility, their complementary determining regions (which confer specificity) are grafted onto human antibody scaffolds that are less immunogenic [3, 4]. However, this process is costly and less adaptable to other proteins. Computational methods might overcome these issues. For instance, Bootwala et al. [5] demonstrated substantial reduction in binding to murine, rabbit, and human polyclonal antibodies. Another strategy for minimizing antibody visibility is employed by Yachnin et al. [6], wherein they aim to decrease the presentation of MHC Class II (MHC-II) epitopes that are crucial for B-cell activation.

In contrast, in the context of gene therapy there is a pressing need of effective MHC Class I (MHC-I) deimmunizers, as illustrated by examples where Cytotoxic T-lymphocytes (CTL) autoimmunity targets hepatocytes transduced with the functional Factor IX gene [7] and examples of unwanted immune responses to the bacterial-derived CRISPR-Cas9 gene editing machinery [8]. In contrast, vaccines, especially those targeting viruses, require pathogen sequences that are particularly MHC-I immune-visible, with epitopes that ideally reflect the diversity found in the viral population. This necessitates the development of computational methodologies for modifying immune visibility concerning the cell's internal antigen presentation process through the MHC-I system, where mRNA therapeutics and vaccines are translated into proteins.

In this article we make the following novel contributions towards the development of the Controlled Amount of Present Epitopes (CAPE) framework for MHC-I visibility modification. (1) We develop `CAPE-XVAE`, a novel machine learning (ML)-based approach that combines the abilities of Variational Autoencoders (VAEs) with reinforcement learning (RL), to generate protein sequences with modified MHC-I immune visibility, (2) We modify the existing physics-based approach from Yachnin et al. [6] that uses the Rosetta *Packer* [9, 10] in a novel fashion to provide it with the ability to reduce MHC-I immune visibility. We call this `CAPE-Packer`. And finally (3) We apply these two novel methods to the HIV Nef protein to compare their ability to modify immune visibility while maintaining stability and similarity of generated synthetic sequences to naturally occurring sequences. We find `CAPE-XVAE` is able to effectively modify the immune-visibility of its generated Nef proteins, while maintaining similarity to observed Nef proteins and stability. `CAPE-Packer` on the other hand, is able to reduce the immune-visibility even further, however does produce proteins, that are very different from the naturally occurring proteins in sequence space.

2 Material and methods

`CAPE-XVAE` is a VAE that represents a protein sequence in a low dimensional latent space, based on which its auto-regressive (AR) decoder is trained to reconstruct the protein sequence. RL is applied to modify the amount of presented epitopes by updating the protein's latent presentation in the direction of increased or decreased MHC-I immune visibility (see `Box1`). In contrast, in `CAPE-Packer` another term is added to the energy function used by the Rosetta *Packer* [9, 10] to allow it to take into account the MHC-I immune visibility for protein design. To examine the effectiveness of the proposed approaches, we apply them to the HIV Nef protein, due to a shorter sequence length of slightly above 200 amino acids (AAs) and the availability of roughly 25 thousand different sequence examples. We assessed the feasibility of our approaches in reducing immune visibility while maintaining stability and similarity to observed Nef proteins.

Important terms used in this paper are presented in `Box1`. The next Subsection will introduce the dataset utilized for our analysis. Following this, we will describe an approximation method we utilized to speed up the peptide presentation assessment. We then proceed to outline the implementation details of our ML approach `CAPE-XVAE`. Afterwards, we delve into the physics-based approach `CAPE-Packer`. Finally, we provide a description of our evaluation process for the generated protein sequences. We often required the corresponding 3D Structure for an AA protein sequence. In these cases we used the online service [1] of the state-of-the-art *ESMFold* [11] language model trained to predict the 3D structure of a protein given its amino acid sequence.

Box 1: Terms and concepts

- **MHC-I immune visibility (also just 'visibility'):** The human immune system is comprised of many components. Our study focuses on the MHC-I pathway which is used by CTL to detect destruction targets. Within most human cells proteins constantly get produced by ribosomes and degraded by proteasomes into smaller bits - so called peptides. Some of these get transferred into the cell's endoplasmic reticulum where a subset will bind to MHC-I proteins (typically 8 to 10 AAs long). These complexes get transported and displayed on the cell surface to be surveilled by CTL.

  MHC-I alleles vary in peptide specificity. To assure the presentation of a broad range of peptides, evolution has resulted in a diverse set of major histocompatibility complex (MHC) proteins being expressed across the human population. In fact, three different main genes are present in humans (HLA-A, HLA-B and HLA-C), allowing each individual to express up to six different MHC-I proteins [12, Chapter 6].

  We define the visibility of a protein to be the number of 8, 9 and 10-mers within a protein sequence, that are predicted to be presented on the cell's surface by the selected subset of MHC-I alleles. Overlapping k-mers are counted separately. Also, if a k-mer is presented by several MHC-I alleles, then each one is independently increasing the visibility number by one. For this study we selected the following common MHC-I alleles to tailor it towards a hypothetical patient: HLA-A*02:01, HLA-A*24:02, HLA-B*07:02, HLA-B*39:01, HLA-C*07:01, HLA-C*16:01.

- **Support sequences:** Often applying an analysis to all natural sequences in the dataset would be prohibitively computationally expensive. To still be able to describe the behaviour of the natural sequences, we once randomly selected 300 representative sequences from the dataset and applied the analysis just to those. In these cases we will refer to the concerned sequences as 'support', while when we refer to all observed sequences we will refer to them as 'natural'.

- **Closest natural sequence:** For a given (source) sequence, *closest natural sequence* refers to the natural sequence in the total dataset with the highest pairwise sequence alignment score (target sequence). If the source sequence is coming from the dataset itself (e.g. a support sequence), the source sequence is excluded from the potential target sequences. Due to the large difference in sequence space, looking at a pairwise sequence alignment between `CAPE-Packer` generated and data sequences is not meaningful. We, therefore, define the *closest natural sequence* for these as the sequence that produced the 3D structure that acted as the input to the `CAPE-Packer` algorithm. We use the Gotoh global alignment algorithm [13] as implemented by the biopython [14] PairwiseAligner (using BLOSUM62, open gap score of -20 and extend gap score of -10).

2.1 Protein dataset

On the 6 June 2023 we downloaded unaligned HIV Nef sequences from the *Los Alamos National Laboratory* HIV Sequence Database [2]. In order to avoid using sequences with sequencing errors, we only considered sequences between 200 and 220 AAs length. Many sequences occur multiple times. During `CAPE-XVAE` training we over-weighted those according to their number of occurrences. We keep the sequences containing 'X' (unknown AA, *complete sequences* do not contain any of those) in the dataset for training. However, these are not considered for the evaluation. We then randomly split up our dataset into the following splits:

[1] https://api.esmatlas.com/foldSequence/v1/pdb/
[2] https://www.hiv.lanl.gov/components/sequence/HIV/search/search.html

Split	Number of Sequences	Complete Sequences
Train	24,026	21,378
Validation	1,334	1,189
Test	1,336	1,189

Table 1: Number of unique sequences in dataset splits

2.2 MHC-Class 1 visibility assessment

For the estimation of MHC-I immune visibility, our study utilizes *netMHCpan-4.1* [15]. This tool represents the latest incarnation in a long line of MHC-I prediction models that goes back to the early 2000s. It operates by taking a peptide sequence and a MHC-I allele (internally represented by a selection of the AAs lining the MHC binding groove) as inputs. The model predicts a score, with higher values indicating greater confidence in the peptide being presented by the MHC-I protein. The *EL-rank* is the score's reversed rank relative to random peptides. The predicted *EL-rank* serves us to assess MHC-I presentation of the peptide. Peptides with an *EL-rank* of below 2% are considered "weak binding" and those with a *EL-rank* of below 0.5% are "strong binding" [3].

As we explored approaches to incorporate the immune-visibility constraint into the protein design process, we found that the approach suggested by Yachnin et al. [6] requires many peptide presentation prediction requests making using *netMHCpan* directly in `CAPE-Packer` infeasible (see Subsection 2.4). To reduce the execution time of the immune modification process in `CAPE-Packer` and `CAPE-XVAE` (we still rely directly on *netMHCpan* during evaluation), we approximated the behavior of *netMHCpan* using a position weight matrix (PWM) based approach. To construct the PWMs, we initially generated a uniform sample of three million random peptides of lengths 8, 9, and 10 AAs. Next, we employed *netMHCpan* to calculate the *EL-rank* for all considered MHC-I alleles. We then selected peptides with an *EL-rank* below 2%. For each position within these peptides, we computed the distribution of AAs, thereby obtaining a PWM for each MHC-I allele and peptide length. Next, we scored all the random peptides using these PWMs. This scoring involved summing up the logarithmic probabilities assigned by the PWM to the AAs at their corresponding positions within the peptide. The scores we can then use to establish rankings for peptides, equivalent to the *EL-rank*.

2.3 CAPE-XVAE

In contrast to `CAPE-Packer`, we built `CAPE-XVAE` up from the ground. The description of its inner workings starts with an introduction to its architecture, followed by a description of the VAE training process before an outline of the RL immune modification procedure. `CAPE-XVAE`'s hyperparameters can be found in the Supplement S.4.

2.3.1 Architecture

`CAPE-XVAE` is based on the VAE architecture [16] and, therefore, comprises an encoder, a latent space, and a decoder.

- **Encoder:** `CAPE-XVAE`'s encoder consists of an embedding layer, a transformer encoder stack (see Supplement S.2) and a block embedding the stack's output into the latent space. Along with a learnt positional encoding, each AA in the input protein sequence x is first encoded as a d dimensional vector using a learnt embedding layer. The encoder stack processes these into a same length sequence of contextual encoded vectors.

  To create an information bottleneck, a cross-attention layer produces a sequence of $L < N$ (currently $L = 1$) latent vectors (each of dimension d) from these contextual encoded vectors. Two linear transformations reduce each latent vector further to $l < d$ dimensions. The first transformation generates a mean vector for the latent's conditional distribution $p(z|x)$, while the second one produces its log variance vector.

- **Latent Space:** A vector z can be sampled from the latent's conditional distribution, if it should be close to the sequence x fed into the encoder (e.g. during training), or from its unconditional distribution $p(z)$ if it should be from the whole distribution of the training set. For the loss function, we assume $p(z)$ to be a standard normal distribution. However, we acknowledge that this assumption may not accurately represent the distribution of natural sequences in latent space. So, when generating new examples, we calculate the mean and covariance matrix of the latent vectors of the training set examples and sample from a Normal distribution with those parameters.

- **Decoder:** As input, it receives the latent vector, and the AAs in the sequences sampled so far - or the whole sequence x during training. It comprises an embedding layer for the AA sequence, a transformer decoder stack (see Supplement S.2) and a head for AA prediction. We embed each sequence element with a learnt token embedding, a learnt positional embedding and a learnt next position embedding. The next-position embedding allows the prediction of protein sequences in arbitrary order (not only in N to C direction). Then, the decoder stack consists of multiple cross-attention blocks. These blocks can attend to the latent vector z generated in the previous step and the portion of the AA sequence generated so far. The sampling is initialized with a start/stop token in the AA sequence and is followed by the element predicted by the previous element in the sequence.

[3] https://services.healthtech.dtu.dk/services/NetMHCpan-4.1/

2.3.2 Training

Training a VAE in conjunction with an AR decoder can be challenging. To address this, we adopted the approach proposed by Li et al.[17] (see Supplement S.1). We fixed the free bits parameter λ as one. So, our training process follows the steps below:

1. **Auto encoder (AE) training:** We trained `CAPE-XVAE` like an AE for 500 epochs (KL divergence weight β in VAE loss function is zero). This focuses training solely on reconstructing the input sequence resulting in very meaningful latent representations.
2. **Reset the decoder**: Afterwards, the decoder is reset to its initialization. This counters the risk of the decoder learning faster than the encoder.
3. **Transition to VAE training:** Over the next 250 epochs the KL-divergence weight in the loss function is increased linearly until it reaches one.
4. **VAE training:** Then we continue training the model for an additional 750 epochs. This phase optimizes the full VAE objective with free bits.

To further increase the decoder's reliance on the latent variable, we train the decoder to sample the protein sequence in random permutations of the ordering - not just N-C direction. Also, teacher-forcing is used during training. As optimizer, we utilize Adam [18] ($\beta_1 = 0.9, \beta_2 = 0.999$) in combination with cosine annealing. To perform a systematic hyperparameter search, we employ Optuna [19] using the tree-structured parzen estimator method. The search objective is to minimize the VAE validation set loss including free bits (see Supplement S.4).

2.3.3 Procedure for designing proteins with modified immune visibility

We reduce or increase the immune visibility of sampled proteins by combining the capabilities of VAEs and RL techniques. VAEs learn to attribute sensible protein sequences to likely points in the latent space. However, as has been seen in [20], only small reductions in immune visibility are likely to be achieved by directly sampling from this latent space. As backpropagation through the decoder's discrete output is not directly possible, we formulate our task as a RL problem (see Supplement S.3) and conduct a directed search in the latent space for vectors of less/more immune visibility.

The state is the protein sequence produced so far. `CAPE-XVAE`'s decoder represents the agent and `CAPE-XVAE`'s next token prediction represents the policy. We also define an immune visibility dependent reward function (see 'Episode loop' below). The episode ends when the decoder produces a stop token. Overall, our modification process follows the following steps also depicted in **Figure 1**:

1. **Initialization:** A random vector $z^{(-1)}$ is sampled in latent space (normally distributed with mean and standard deviation estimated using the latent vectors of the naturally occurring training data sequences) and decoded to retrieve an initial sequence $s^{(0)}$. This is fed into the encoder to retrieve an initial latent vector $z^{(0)}$ and a standard deviation σ around it. To ensure that the deimmunized latent variable stays in a likely area of the latent space, we force it to be within a certain range $m \cdot \sigma$ around the $z^{(0)}$. To do so, we use an auxiliary latent vectors $w^{(j)}$ in an auxiliary latent space W, which is mapped into the sphere with radius $m \cdot \sigma$ in latent space surrounding $z^{(0)}$. We use Manhattan distance and m is currently set to 2.
2. **Episode:** At the beginning of each episode j the current auxiliary $w^{(j)}$ is converted to a latent variable $z^{(j)}$, which remains constant throughout the episode. The decoder uses this to sample 64 AA sequences in parallel in N-C direction. The temperature used for this is linearly decreased from 1.2 to 0.1 over the deimmunization phase. The gradient of the log-probability of the sampled AA with respect to the auxiliary latent variables is calculated and saved. Also, for each added AA the reward function is called and its result saved as this step's reward. In addition to the current state, e.g. MGGKWSKSSSEAKWPAVRERMRRTEPADG), and the current action (e.g. V) the **reward function** receives the selected MHC-I alleles as input. As most MHC-I epitopes have length 8, 9 or 10 AAs, the terminal 8-, 9- and 10-mers of the combined sequence (e.g. MRRTEPADGV, RRTEPADGV, RTEPADGV) are examined using the PWM classifier for MHC-I presentation (see Subsection 2.2). Each presented peptide reduces (deimmunizations) or increases (immunizations) the reward by one. The sum over all MHC-I alleles and peptide lengths, represents the reward at the corresponding step in the trajectory, penalizing or rewarding presented peptides as soon as they are fully sampled - but not at all before.
3. **Updating auxiliary latent variables:** At the end of each episode j, the auxiliary latent variable $w^{(j)}$ is updated to $w^{(j+1)}$ using the REINFORCE algorithm (see Supplement S.3). As baseline, we use the average of the future rewards G for each step. Then the algorithm goes back to step 2 until 100 updating steps have been completed.

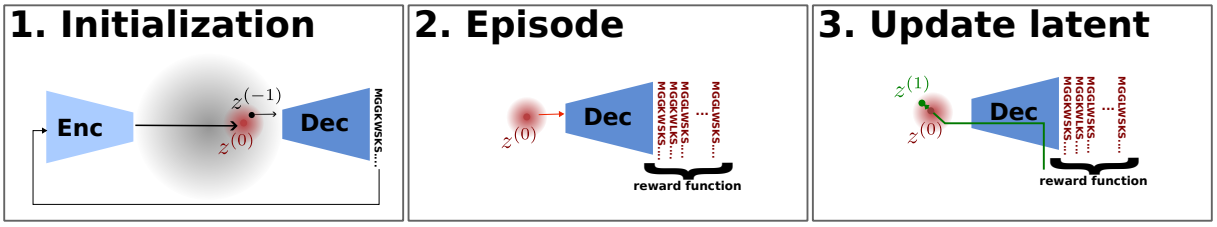


Figure 1: Steps followed during the immune visibility modification process

2.4 CAPE-Packer

The implementation of `CAPE-Packer` is based on the framework implemented by Yachnin et al. [6] that modifies Rosetta's energy term to discourages the Rosetta *Packer* from using MHC-II epitopes.

Rosetta is a comprehensive software suite used for modeling and analyzing protein structures. One of its key algorithms is the *Packer*, which serves as a side-chain optimization tool. It identifies the energy minimizing rotamers for a given sequence and fixed backbone structure (pose) [21]. Additionally, the *Packer* can be employed for sequence design by considering different side-chain AA identities. In fact, this is its standard behaviour and the *Packer* designs at every position, allowing for all 20 standard amino acid rotamers. However, a "resfile" can be used to specify a position dependent subset of eligible rotamers. The space of possible rotamer combinations expands rapidly, making enumeration infeasible. Therefore, Monte Carlo simulation is utilized. Starting from a random initialization, random changes are made to the rotamers. If the change decreases the energy, it is always accepted. Conversely, if it increases the energy, acceptance is stochastic with a probability that decreases with the energy [21].

Yachnin et al. [6] have suggested an additional term for this energy function, that takes into account the number of peptides in the protein that get presented on the cell's surface. In their case, they used MHC-II, and we adapted their tool towards MHC-I. Their method only considers 9-mer peptides and out of the box it is only implemented using a PWM approach as well as a lookup database. While they mention that they have experimented with a tensor flow based presentation predictor, no implementation of this is known to us. In fact Mulligan [21] state that "In order to be compatible with the packer, an energy term must either be residue-level pairwise-decomposable, or must otherwise be very fast to compute and update as rotamer substitutions are considered". Therefore, we think the reason for the lack of a more advanced predictor backbone is, that the Rosetta *Packer* requires many, many score function evaluations - and, so, peptide presentation evaluations. For a short 203 amino acid long Nef sequence, we counted roughly 42 million peptide evaluation requests. This has to be multiplied by the number of considered MHC alleles to arrive at the actual number of predictor calls. Using one of the modern and larger MHC-I presentation prediction state-of-the-art models - like *MHCflurry 2.0* [22] or *netMHCpan 4.1* [15] - is, therefore, very time intensive.

We implemented a new Rosetta class `MHCEpitopePredictorClient` that inherits from `MHCEpitopePredictor` and receives 9-mer presentation evaluation requests from Yachnin et al.'s framework and forwards them to a server that we also implemented. This server relies on the same PWM MHC-I presentation predictor as `CAPE-XVAE` (see Subsection 2.2), and returns the number of MHC-I alleles (from the predefined subset) that are predicted to present the peptide. The `MHCEpitopePredictorClient` object then returns this number to the framework built by Yachnin et al., which integrates it into the energy function used by the Rosetta *Packer* that ultimately designs the protein.

As mentioned above, the *Packer* needs 3D structure of the protein as input and searches for a corresponding energy minimizing AA sequences. To select the input 3D structure, we first generated deimmunized CAPE-XVAE sequences and then found their *closest natural sequence*. The *ESMfold* predicted 3D structure of these were then used as inputs to `CAPE-Packer` for deimmunization. We design at every AA position in the sequence, without any restrictions to admissible AA.

2.5 Evaluation

We assessed the generated sequences using three criteria: immune visibility, similarity to natural sequences and stability. We describe each of the criteria below.

2.5.1 Visibility

To assess how MHC-I immune visible (see `Box1`) the protein sequences are to CTL, we used *netMHCpan-4.1* [15]. We used the same MHC-I alleles during the evaluation step as we had used for the modification process (HLA-A*02:01, HLA-A*24:02, HLA-B*07:02, HLA-B*39:01, HLA-C*07:01, HLA-C*16:01). Also the way we assess visibility is the same as in the modification process' reward function (count the number of peptides being predicted to be presented). For readability, we sometimes use the term *visibility* to refer to MHC-I immune visibility.

2.5.2 Similarity

We assess similarity of a generated sequence to its *closest natural sequence* (see `Box1`). The root mean squared distances (RMSDs) between the two structures was calculated using *TMalign* [23]. This software also calculates the TMscore

which ranges between zero and one. A score of one signifies a perfect match while scores below 0.2 are associated with randomly chosen unrelated proteins. Scores higher than 0.5 assume generally the same fold in SCOP/CATH [4]. To assess whether there is a similarity in parts of the sequences to the data sequences, we checked the k-mers in the sequence against all k-mers occurring in the data sequences. The percentage that is also present in the data sequences we define as the *k-mer similarity*. A k-mer similarity of particular interest is the 9-mer since most presented peptides have 9 AA length. So, the number of naturally occurring 9-mers is particularly relevant for vaccine design, where we try to design sequences that more broadly capture epitopes across variants of a protein. For a particular generated k-mer, we also define *k-mer precision* as the percentage of the natural sequences that include this k-mer. In contrast, the *k-mer recall* between a generated and natural sequence is the percentage of k-mers in the natural sequence that are also present in the generated sequence.

2.5.3 Stability

To assess the stability of our generated protein sequences in comparison to their *closest natural sequence* (see `Box1`) we used *FoldX Stability* [24] and *DDGun* [25]. There is a difference between the Gibbs free energy of a protein in its folded state and its unfolded state (ΔG). In general, the lower this is, the more stable the protein will be. We use *FoldX Stability* to calculate the energy in the protein structures. An alternative approach is to directly estimate the impact of differences between a sequences on their stability. In this case, we are interested in the difference in ΔG between the wild type and the mutated protein (in this paper we use $\Delta\Delta G = \Delta G_{mutant} - \Delta G_{wildtype}$). Unlike some other approaches that rely on structural information, *DDGun* only requires sequence inputs and utilizes evolutionary information provided by *HHblits*, which scans Uniclust30. It is an untrained algorithm that estimates the impact of single- and multiple substitutions on ΔG. Remarkably, *DDGun* achieves competitive performance to state-of-the-art approaches, as measured by correlation coefficients between predicted and measured $\Delta\Delta G$, even without relying on structural data [25]. This characteristic makes it particularly relevant for our application, where access to crystal structure-based information is limited, which might make *FoldX Stability* results less reliable. For the evaluations, we considered sequences where the only mutations to the *closest natural sequence* where substitutions. This is because *DDGun* can only handle substitutions.

2.6 2D sequence space visualizations

In order to visualize the distribution of sequences in two dimensional space, we used the dimension reduction technique t-SNE. First we construct a 300 dimensional representation for each sequence. Each dimension represents the distance between the sequence and one of the *support sequences* (see `Box1`). We compute this distance by first calculating the pairwise alignment score (same alignment algorithm as for finding the *closest natural sequence*) of the sequence with the *support sequence*. The distance is then defined as the difference between this alignment score and the alignment score achieved by aligning the sequence with itself. To further reduce the dimension of these vectors to only two, we use t-SNE with a perplexity of 30 and running for 500 iterations. We have observed similar results for a broad range of perplexity numbers.

[4] https://seq2fun.dcmb.med.umich.edu//TM-align/

3 Results and Discussion

We present the results of the experiments carried out using the `CAPE-XVAE` and `CAPE-Packer` approaches with respect to MHC-I immune visibility (see `Box1`), similarity to the observed natural sequences, and the stability of the produced proteins.

We first explored the landscape of naturally occurring sequences and those produced by `CAPE-XVAE` and `CAPE-Packer` (**Figure 2**, see Subsection 2.6). Notably, the immune modified sequences produced by `CAPE-XVAE` are dispersed across the natural sequence space in **Figure 2ab** and do not aggregate in any particular region. The RL modification step significantly alters the sequence positions (arrows in **Figure 2ab**). Starkly, the sequences produced by `CAPE-Packer` aggregate distinctly outside the natural background distribution of sequences. In fact these `CAPE-Packer` sequences are drastically different from naturally occurring sequences (**Figure 2c**), despite appearing close in the t-SNE plot (**Figure 2ab**) which is a cluster preserving but not distance preserving dimensionality reduction technique.

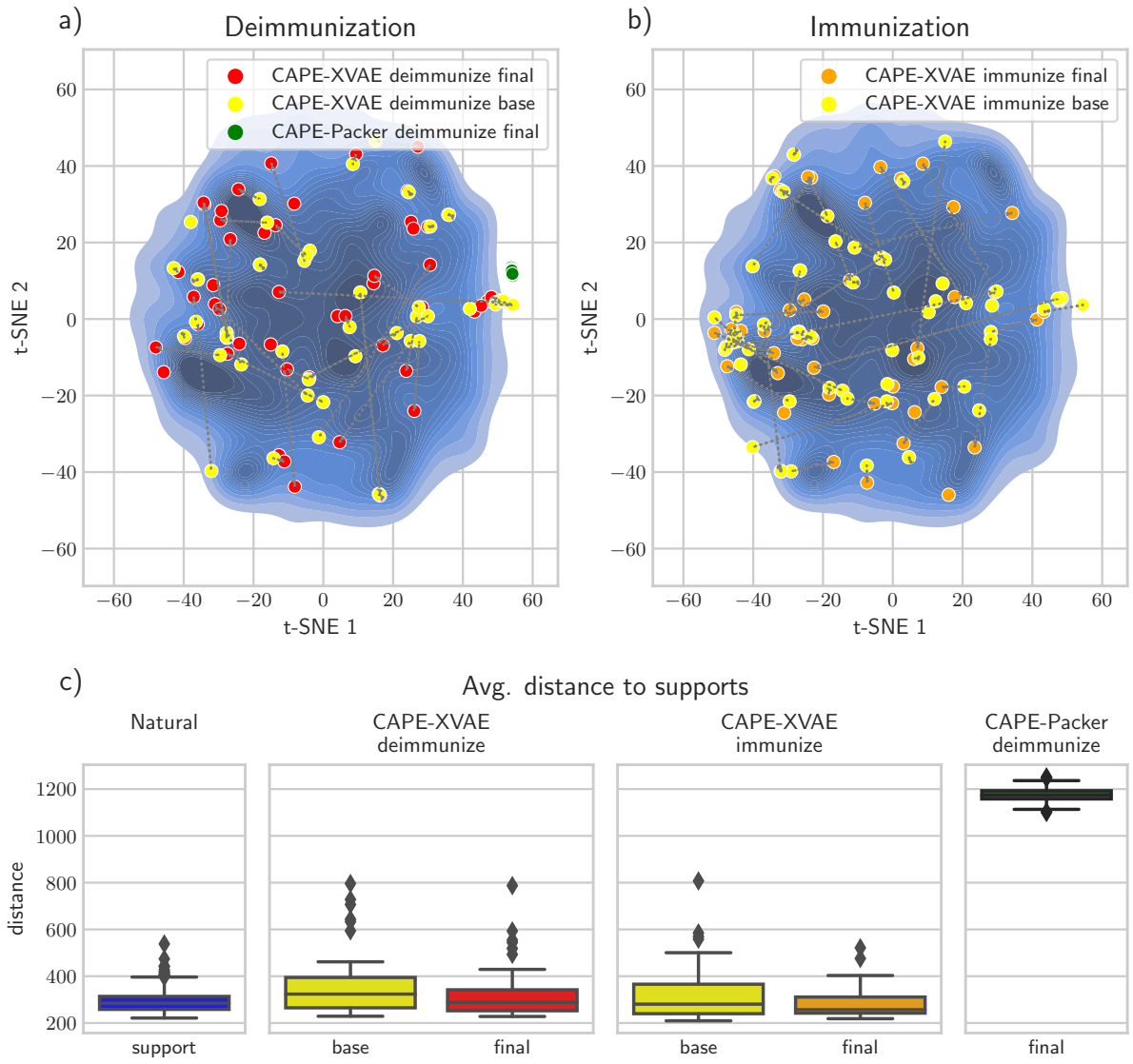


Figure 2: **Evolution and displacement of sequences during deimmunization and immunization processes.** The density of the natural sequences used to train the models is depicted blue in the background. The yellow dots correspond to the initially sampled sequences obtained through `CAPE-XVAE` during the modification process. The arrows starting in the yellow dots show the adjustment performed by the immune-modification process. (a) Shows the deimmunization results. The red dots are the finally obtained deimmunized sequences by `CAPE-XVAE`. In contrast, the green dots are the finally obtained deimmunized sequences by `CAPE-Packer`. (b) Immunization results are displayed. The orange dots show the finally obtained immunized sequences by `CAPE-XVAE`. (c) The distribution of the average distance to the 300 support sequences (see `Box1`). Colors correspond to the other plots

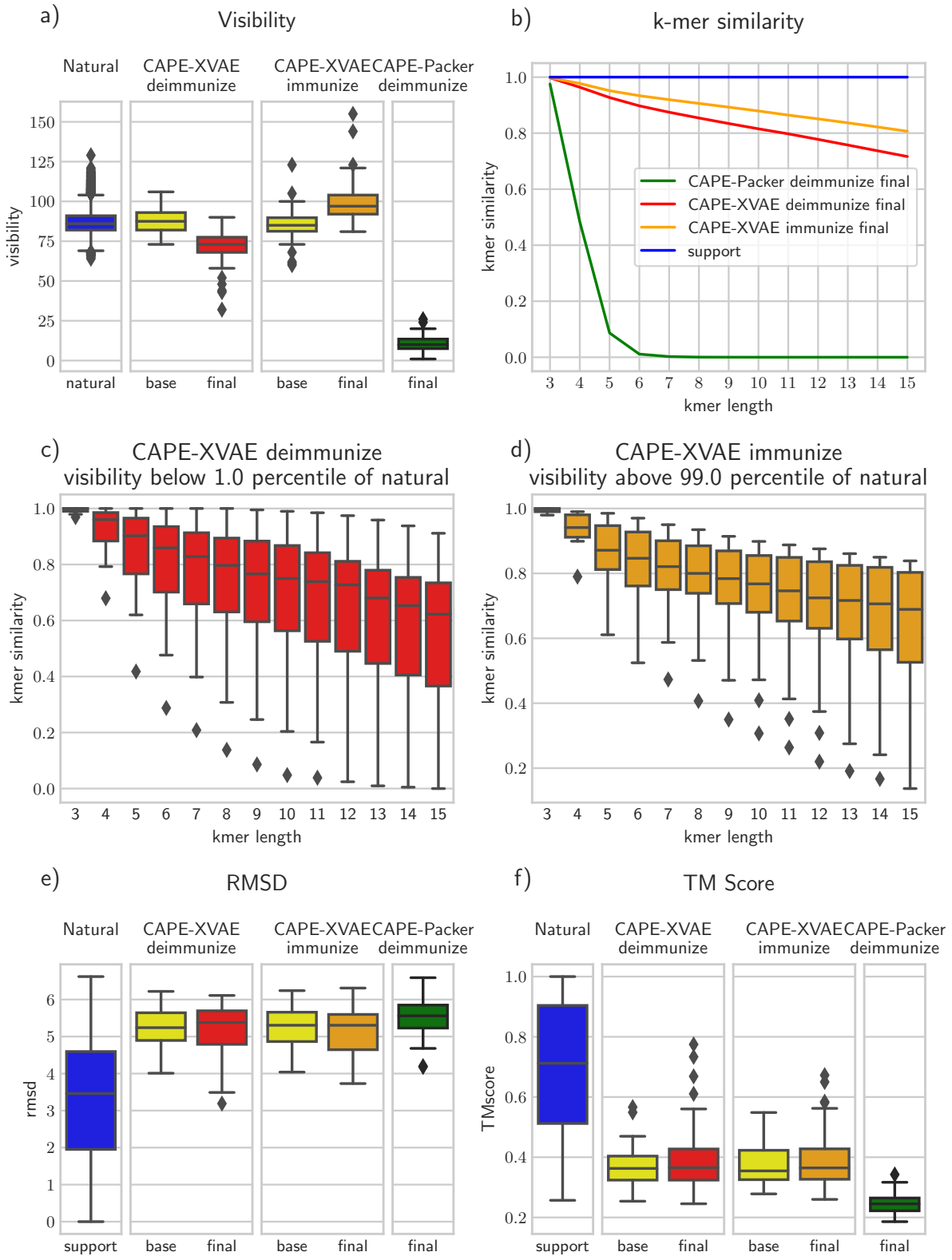


Figure 3: **Altering MHC Class I immune visibility while maintaining natural sub-sequence characteristics.** (a) **Visibility:** Shows the distribution of the visibility of the various sequence sets. The visibility is calculated equivalently to how the *reward function* is calculated for the RL modification process (See Subsection 2.3.3) (b) **K-mer Similarity:** Dependent on k-mer length, this chart shows the average proportion of k-mers with that length in the sequences that can also be found in the natural sequence data. One means that all k-mers can be found in the data and zero means that none of the k-mers can be found in the data. (c,d) Boxplots show the k-mer length dependent distribution of k-mer similarity for the `CAPE-XVAE` modified sequences that are either in the bottom (de-immunized) (c) or top (d) percentile (immunized) of immune-modulated sequences. (e,f) Show the distribution of the RMSD and TM Score as calculted by *TMalign* [23] between the sequences and their *closest data sequence*. Color scheme continues from **Figure 2**

The MHC Class I system recognizes short sub-sequences of a protein, so we developed a similarity metric to the training set in terms of short peptides (k-mers) up to length 15 (see Subsubsection 2.5.2). Using this metric, we explored the balance our protein design methods strike when altering MHC Class I immune visibility while keeping a close resemblance to natural functional sequences. Our goal was to understand if the two strategies introduce natural looking sequences (k-mer similarity - see Subsubsection 2.5.2).

We find that `CAPE-XVAE` and `CAPE-Packer` are indeed able to alter MHC-I immune visibility (see `Box1`) when compared to naturally occurring sequences. While `CAPE-Packer` can produce sequences that are much lower in immune visibility (**Figure 3a**), this comes at the cost of lower k-mer similarity (**Figure 3b**). Indeed, while `CAPE-XVAE` is heavily incorporating k-mers from the dataset, `CAPE-Packer` incorporates nearly only k-mers outside the naturally occurring sequences from 6-mers onward. Even in the case of those `CAPE-XVAE` generated sequences that were visible above the top (**Figure 3c**) and below the bottom (**Figure 3d**) percentile of immune-visible natural sequences, over 60 percent of k-mers were to be found in the dataset of naturally occurring sequences.

This conservative behavior of `CAPE-XVAE` may be beneficial in vaccine design, where protein sequences should be representative of the population of sequences (i.e. viruses) used to train the model. Vaccines would ideally sample the breadth of MHC-I epitopes (8, 9 and 10-mers) observed in the pathogen.

Even for de-immunization tasks, a constraint to remain within natural sequences may increase the chances that complex nuances of protein function are retained in the synthetic sequence. Models like `CAPE-Packer`, that are only cognizant of protein fold, without understanding the grammar of protein function can push the protein's sequence far outside of naturally looking sequences. Even from a structural perspective, `CAPE-XVAE` sequences seem to be slightly more similar to the training sequence distribution by RMSD (**Figure 3e**) and TM Scores (**Figure 3f**) to their *closest natural sequences* (see `Box1`). As can be seen in **Figure 3e**, the RMSDs of the `CAPE-XVAE` and `CAPE-Packer` generated sequences are within the distribution of the data, although on the upper end. In contrast, the TM scores of `CAPE-Packer` are below what we can find within the data.

Next, **Figure 4** shows how our measures of similarity and stability change dependent on visibility. In general, we find that the further we move away from the data distribution, the less similar and stable the generated sequences become. However, we find that in particular the immunized sequences produced by `CAPE-XVAE` manage to retain a fairly high 9-mer similarity even for highly visible proteins. In contrast, 9-mer similarity is not present for any of the `CAPE-Packer` generated sequences. RMSD of all our generated sequences are in the area of five Angstrom. However, it is questionable whether a folding predictor like *ESMfold*, that is trained on natural sequences, is really capable of predicting the 3D structure of artificially generated sequences. *DDGun* only relies on sequence and not structure and is capable of handling multiple substitution mutations. However, there is rarely any AA the same in a `CAPE-Packer` generated sequence as in its *closest natural sequence*. Because of this we also used the physics base *FoldX* to directly calculate the Energy of the structure. In general, these seem to be quite high. This points towards problems with the PDB structures. However, since we know that the observed natural sequences (supports) are stable, we are more interested in a relative treatment. We see that the energy is very similar between the observed data sequences (support) and the `CAPE-XVAE` generated sequences. In contrast, the energy seems to be higher in the `CAPE-Packer` generated sequences and also span a larger range - pointing towards less stable structures.

In the context of vaccine design, chimeric sequences that immunize against more than one variant of a protein could provide better protection to the population. `CAPE-XVAE`, proposes sequences similar to those natural sequences used to train it and we were interested to understand how diversely k-mers predicted to be immune-visible were sampled from the natural sequences. **Figure 5** explores the ability of a representative `CAPE-XVAE` immunized sequence to capture the naturally occurring k-mer and epitope diversity. We focus here only on 9-mers and predicted 9 AAs long MHC-I epitopes, which is the most common length of an MHC Class I epitope. We selected a `CAPE-XVAE` sequence with immune visibility score of 114 - which corresponds to the 99.9 percentile of the immune visibility score in naturally occurring sequences. We find that most of the k-mers in this immunized generated sequence can indeed be found in the natural sequences (**Figure 5ab**). In particular only 4 of the visible 9-mers were not at all present in the natural sequences (**Figure 5b**). Importantly, 99.5% of natural sequences have at least one visible 9-mer epitope that is also present in our selected sequence (**Figure 5c**), highlighting the ability of this sequence to present epitopes of a broad range of viral variants to the immune system. In fact the average visible 9-mer recall across all natural sequences is 9% for the synthetically generated sequence, while this number is only 6.4% on average for all 27 natural sequences with a visibility between 111.5 and 116.5. Thus, we contend that this approach is highly compatible with vaccine development due to the ability of language models to tap into the array of natural viral variants, and integrate them into synthetically constructed sequences that may better reflect the diversity of the variants in question than individual natural sequences.

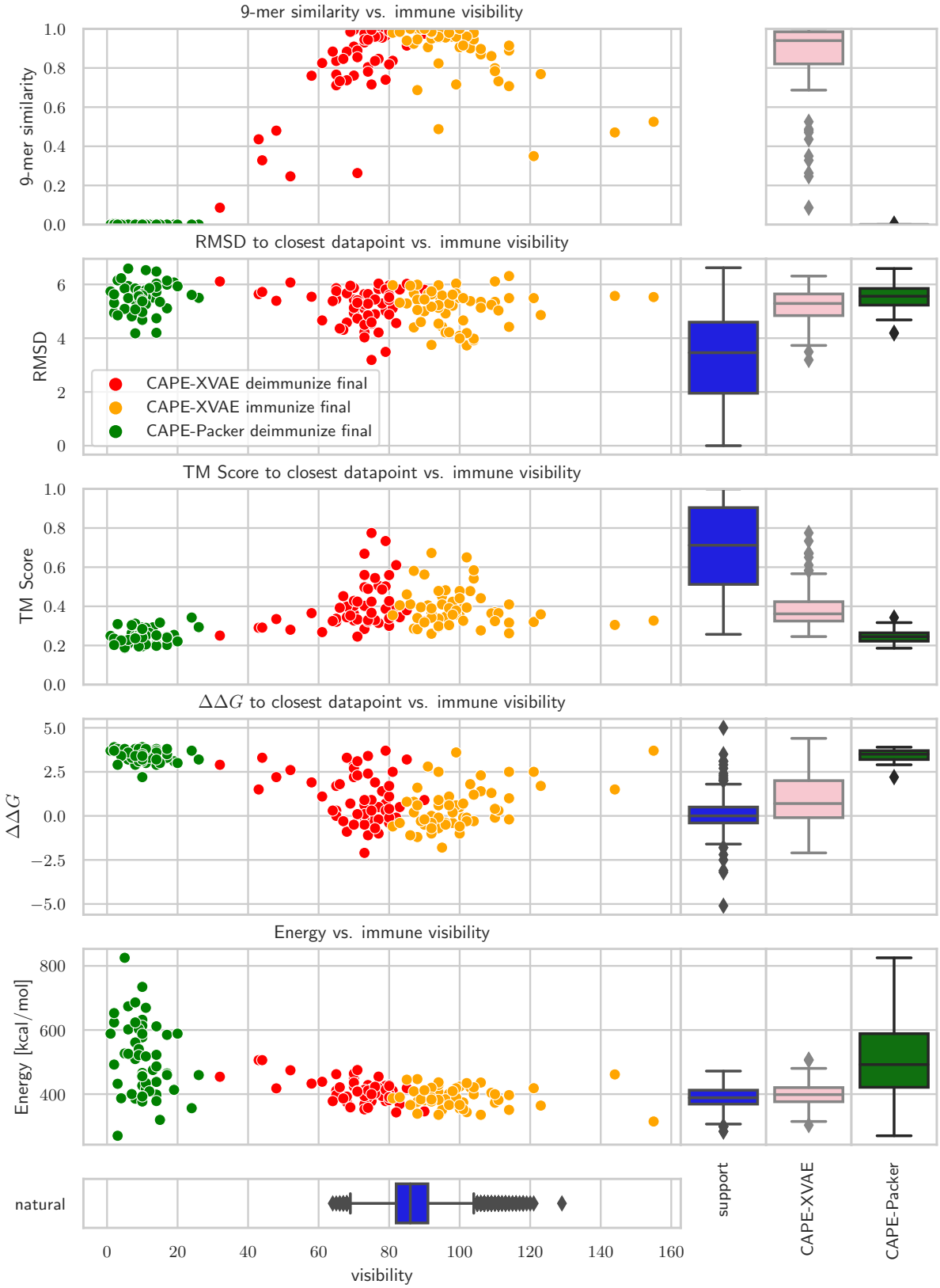


Figure 4: **Assessing similarity and stability dynamics in protein design: A comparative study of visibility-dependent trends in CAPE-XVAE and CAPE-Packer Generated Sequences.** In each row of this chart, we show how the natural data sequences compare to `CAPE-XVAE` and `CAPE-Packer` generated sequences with respect to one of the following criteria: 9-mer similarity, RMSD, TM Score, $\Delta\Delta G$ and Energy. Each dot in the left column represents a single sequence - the same color coding as in Figure 2 applies. The x-axis shows the immune visibility. The three boxplots per row on the right display the distribution within the compared values for the support as well as the two model generated sequences. The horizontal boxplot on the bottom, shows the distribution of visibility within the data.

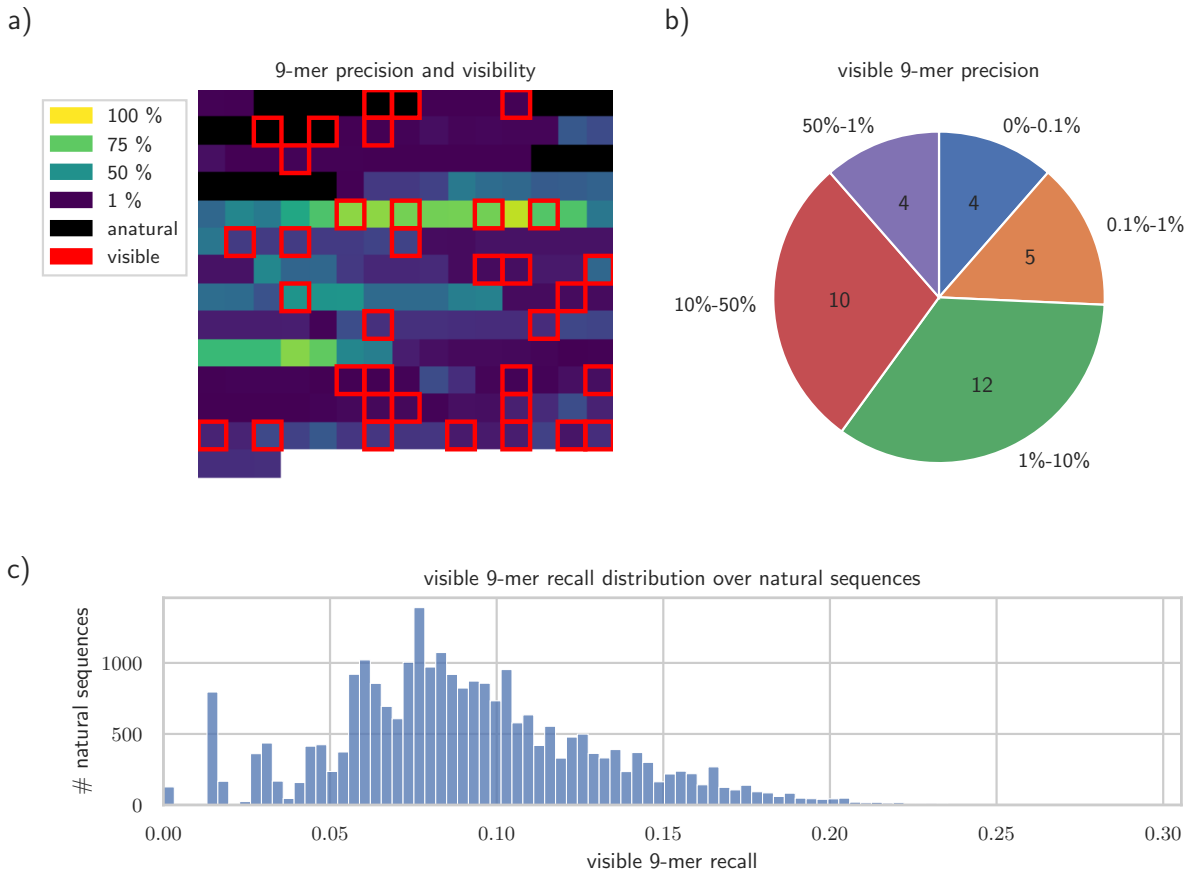


Figure 5: **Language models in vaccine design: borrowing from the diversity of existing protein variants**. Vaccines must contain real epitopes existing across variants of a virus. This figure shows an epitope map of an immunized sequence produced by `CAPE-XVAE`, which has a predicted visibility score of 114. We investigate where each epitope is derived from within the natural distribution of input sequences. (a) We present 9-mer peptides in the Nef protein sequence using a sliding window approach advancing 1 amino acid at a time. The N-Terminal 9-mer is top left progressing one amino acid at a time to the C-Terminal 9-mer on the bottom right. The colors signify how often these 9-mers occur among the natural sequences (see *k-mer precision* in 2.5.2). A red box signifies that this 9-mer is predicted to be presented at least by one MHC allele. (b) The inner labels of the pie chart are the number of presented 9-mers [red boxes in (a)] and the outer labels describe the percentage of natural sequences that contain this epitope (c) For all natural sequences we identify the visible 9-mers and calculate the percentage of those that are present in the example immunized sequence, demonstrating that the language model samples broadly from the existing Nef viral protein diversity. The histogram shows the distribution of this number over the natural sequences (see *k-mer recall* in 2.5.2)

4 Conclusions

Designed protein-based medicines have the potential to change healthcare by serving as powerful vaccines and therapeutics for a variety of diseases. It is important that these medicines do not accidentally set off an immune response unless designed to do so, as with vaccines. In this work we have developed `CAPE-XVAE` a language model that learns from the naturally occurring sequences and proposes new synthetic constructs that are more visible to the immune system when needed, and stealthy when not. We focused on optimizing the visibility to Cytotoxic T-cell via the MHC Class I pathway. We also develop a de-immunization model ignorant of the natural sequence space of the target protein, `CAPE-Packer`, and contrast it to our language model approach. Designing proteins within the boundaries of natural sequences is crucial for two main reasons. Firstly, it helps in preserving the known and potentially undiscovered functions of the protein being designed. Secondly, for vaccines, it potentially allows to reflect the diversity of epitopes found in viral protein sequence variants making escape mutation more unlikely.

In `CAPE-XVAE`, we implemented a novel architecture that introduces a low dimensional latent space in between the traditional transformer encoder and decoder. With `CAPE-Packer`, we also implemented a physics-based approach that minimizes an energy function that also includes a term for the immune visibility of the sequences. This is implemented as a novel add-on to a MHC-II deimmunization protocol developed by Yachnin et al. [6].

We find that `CAPE-XVAE` is able to sample protein sequences with modified immune visibility compared to naturally occurring ones. However, the natural sequence space of our target proteins (over 20,000 sequences; Table 1), does not

provide sufficient sequence diversity to completely hide the protein from the immune system in our example case. The sequences produced by `CAPE-XVAE` are more similar in structure to the nearest naturally occurring sequence as judged by TM-score and RMSD than those produced by `CAPE-Packer`. Even further, sequences generated by our language model incorporated a large proportion of short sub-sequences that could themselves be found in the training set (k-mer similarity). This is a particularly important property from the perspective of ensuring conservation of local functional constraints, and is particularly useful for vaccine design. In an example designed protein, we saw in Figure 5, that indeed these epitopes were broadly sampled, covering the bulk of the known sequence space for the protein in question.

In contrast, `CAPE-Packer` which is not cognizant of the natural sequence space was indeed able to produce de-immunized sequences with less immune visibility than `CAPE-XVAE`. However, these sequences were exceedingly divergent in sequence space as measured by our k-mer similarity metric. Additionally, this method relies on the availability of a good 3D protein structure which is not always available, for example if designing disordered proteins. On the other hand, `CAPE-XVAE` requires a diverse set of sequences to learn from. As a limitation to our study, we note that our evaluation approach relies on the state-of-the-art *ESMfold* folding prediction model to produce accurate structures. It is unclear whether in particular for `CAPE-Packer`, which produces very divergent sequences, the current folding prediction models which rely on learning from naturally occurring sequences will remain accurate.

Based on these observations, we conclude that approaches like `CAPE-XVAE` that rely on learning solely the distribution of natural protein sequences may struggle in their efforts to reduce immune visibility. Indeed, if the subsequences that need to be introduced do not exist in the training data, it can be a challenge for these methods to propose them. However, they may hold value to increasing immune visibility in a natural way that is particularly useful in vaccine design. All together, `CAPE-XVAE`, may represent the starting point for a drastically different approach to vaccine development compared to that employed during the SARS-Cov-2 pandemic, where vaccines were generated against specific variants. The concept will need validation, and this will be the subject of future studies. Indeed, designed sequences that borrow from the existing diversity of the virus may provide better, more universal vaccines into the future, compared to those that use just one strain of a virus. In contrast, we saw that physics-based approaches like `CAPE-Packer` tend to result in proteins that are very different in sequence space from natural proteins. They might, therefore, have issues mimicking protein functions that could be unknown to the designer at the time of engineering a protein therapeutic. For example, in attracting similar post-translational modification dynamics, or interacting with a diversity of proteins.

Because of this, we think that new ML-based approaches will need to be developed that implement a tunable and hybrid approach. Language models must utilize much more data, utilizing also information from other proteins and must also be able to diverge by a defined and tunable amount based on physics potentials. Then, similarly to large language models, they will have their outputs modified in a desirable way using RL methods.

5 Acknowledgements

This work was supported by the United Kingdom Research and Innovation (grant EP/S02431X/1), UKRI Centre for Doctoral Training in Biomedical AI at the University of Edinburgh, School of Informatics. For the purpose of open access, the author has applied a creative commons attribution (CC BY) licence to any author accepted manuscript version arising.

This project was supported by the Royal Academy of Engineering and the Office of the Chief Science Adviser for National Security under the UK Intelligence Community Postdoctoral Research Fellowship programme.

J.A. Alfaro was supported by (i) United Kingdom Research and Innovation (grant EP/S02431X/1), (ii) the project 'International Centre for Cancer Vaccine Science' which is carried out within the International Agendas Program of the Foundation for Polish Science, cofinanced by the European Union under the European Regional Development Fund, and (iii) The JunG Grant, awarded to Dr. Javier Alfaro at the University of Gdansk. The authors would like to thank 'CI-TASK, Gdansk', and the 'PL-Grid Infrastructure, Poland' for providing their hardware and software resources. Dr. Rajan was supported by the KATY Consortium H2020-SCI-FA-DTS-2020-1.

6 Author contributions

Hans-Christof Gasser: Conceptualization, Methodology, Software, Validation, Writing - Original Draft, Writing - Review & Editing, Visualization **Diego Oyarzun:** Conceptualization, Writing - Review & Editing, Visualization, Supervision **Ajitha Rajan:** Conceptualization, Writing - Review & Editing, Visualization, Supervision **Javier Alfaro:** Conceptualization, Writing - Review & Editing, Visualization, Supervision

During the preparation of this work the author(s) used ChatGPT in order to increase the readability of the text. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] Moderna. *mRNA Medicines We Are Currently Developing*. 2023. URL: https://www.modernatx.com/en-US/research/product-pipeline (visited on 07/07/2023).

[2] Léa V. Zinsli et al. "Deimmunization of Protein Therapeutics – Recent Advances in Experimental and Computational Epitope Prediction and Deletion". In: *Computational and Structural Biotechnology Journal* 19 (Jan. 2021), pp. 315–329. URL: https://www.sciencedirect.com/science/article/pii/S2001037020305493 (visited on 01/09/2023).

[3] Yoonjoo Choi, Karl E. Griswold, and Chris Bailey-Kellogg. "Structure-Based Redesign of Proteins for Minimal T-cell Epitope Content". In: *Journal of Computational Chemistry* 34.10 (2013), pp. 879–891. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23213 (visited on 01/10/2023).

[4] Yoonjoo Choi et al. "EpiSweep: Computationally Driven Reengineering of Therapeutic Proteins to Reduce Immunogenicity While Maintaining Function". In: *Computational Protein Design*. Ed. by Ilan Samish. Methods in Molecular Biology. New York, NY: Springer, 2017, pp. 375–398. URL: https://doi.org/10.1007/978-1-4939-6637-0_20 (visited on 01/10/2023).

[5] Ali Bootwala et al. "Protein Re-Surfacing of E. Coli L-Asparaginase to Evade Pre-Existing Anti-Drug Antibodies and Hypersensitivity Responses". In: *Frontiers in Immunology* 13 (2022). URL: https://www.frontiersin.org/articles/10.3389/fimmu.2022.1016179 (visited on 01/09/2023).

[6] Brahm J. Yachnin et al. "MHCEpitopeEnergy, a Flexible Rosetta-Based Biotherapeutic Deimmunization Platform". In: *Journal of Chemical Information and Modeling* 61.5 (May 2021), pp. 2368–2382. URL: https://doi.org/10.1021/acs.jcim.1c00056 (visited on 02/16/2023).

[7] Gary C. Pien et al. "Capsid Antigen Presentation Flags Human Hepatocytes for Destruction after Transduction by Adeno-Associated Viral Vectors". In: *The Journal of Clinical Investigation* 119.6 (June 2009), pp. 1688–1695. URL: https://www.jci.org/articles/view/36891 (visited on 03/05/2023).

[8] Julie M. Crudele and Jeffrey S. Chamberlain. "Cas9 Immunity Creates Challenges for CRISPR Gene Editing Therapies". In: *Nature Communications* 9.1 (Aug. 2018), p. 3497. URL: https://www.nature.com/articles/s41467-018-05843-9 (visited on 03/05/2023).

[9] Andrew Leaver-Fay et al. "Chapter Nineteen - Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules". In: *Methods in Enzymology*. Ed. by Michael L. Johnson and Ludwig Brand. Vol. 487. Computer Methods, Part C. Academic Press, Jan. 2011, pp. 545–574. URL: https://www.sciencedirect.com/science/article/pii/B9780123812704000196 (visited on 07/07/2023).

[10] Sarel J. Fleishman et al. "RosettaScripts: A Scripting Language Interface to the Rosetta Macromolecular Modeling Suite". In: *PLOS ONE* 6.6 (June 2011), e20161. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0020161 (visited on 07/07/2023).

[11] Zeming Lin et al. *Evolutionary-Scale Prediction of Atomic Level Protein Structure with a Language Model*. Oct. 2022. URL: https://www.biorxiv.org/content/10.1101/2022.07.20.500902v2 (visited on 01/07/2023).

[12] Kenneth Murphy and Casey Weaver. *Janeway's Immunobiology*. Ninth. Garland Science, 2017.

[13] Osamu Gotoh. "An Improved Algorithm for Matching Biological Sequences". In: *Journal of Molecular Biology* 162.3 (Dec. 1982), pp. 705–708. URL: https://www.sciencedirect.com/science/article/pii/0022283682903989 (visited on 06/01/2023).

[14] Peter JA Cock et al. "Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics". In: *Bioinformatics (Oxford, England)* 25.11 (2009), pp. 1422–1423.

[15] Birkir Reynisson et al. "NetMHCpan-4.1 and NetMHCIIpan-4.0: Improved Predictions of MHC Antigen Presentation by Concurrent Motif Deconvolution and Integration of MS MHC Eluted Ligand Data". In: *Nucleic Acids Research* 48.W1 (July 2020), W449–W454. URL: https://doi.org/10.1093/nar/gkaa379 (visited on 04/05/2021).

[16] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [Cs, Stat]*. May 2014. URL: http://arxiv.org/abs/1312.6114 (visited on 02/07/2022).

[17] Bohan Li et al. "A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3601–3612. URL: https://www.aclweb.org/anthology/D19-1370 (visited on 02/15/2023).

[18] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). URL: http://arxiv.org/abs/1412.6980 (visited on 10/22/2020).

[19] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *arXiv:1907.10902 [cs, stat]* (July 2019). URL: http://arxiv.org/abs/1907.10902 (visited on 03/15/2022).

[20] Dominic Phillips et al. "Generating Immune-aware SARS-CoV-2 Spike Proteins for Universal Vaccine Design". In: *Proceedings of the 1st Workshop on Healthcare AI and COVID-19, ICML 2022*. PMLR, July 2022, pp. 100–116. URL: https://proceedings.mlr.press/v184/phillips22a.html (visited on 07/25/2022).

[21] Vikram Khipple Mulligan. *Optimizing Side-Chains: Introduction to the Packer*. June 2016. URL: https://www.rosettacommons.org/demos/latest/tutorials/Optimizing_Sidechains_The_Packer/Optimizing_Sidechains_The_Packer (visited on 06/07/2023).

[22] Timothy J. O'Donnell, Alex Rubinsteyn, and Uri Laserson. "MHCflurry 2.0: Improved Pan-Allele Prediction of MHC Class I-Presented Peptides by Incorporating Antigen Processing". In: *Cell Systems* 11.1 (July 2020), 42–48.e7. URL: https://www.sciencedirect.com/science/article/pii/S2405471220302398 (visited on 04/05/2021).

[23] Yang Zhang and Jeffrey Skolnick. "TM-align: A Protein Structure Alignment Algorithm Based on the TM-score". In: *Nucleic Acids Research* 33.7 (Apr. 2005), pp. 2302–2309. URL: https://doi.org/10.1093/nar/gki524 (visited on 03/31/2023).

[24] Joost Schymkowitz et al. "The FoldX Web Server: An Online Force Field". In: *Nucleic Acids Research* 33.suppl_2 (July 2005), W382–W388. URL: https://doi.org/10.1093/nar/gki387 (visited on 06/15/2023).

[25] Ludovica Montanucci et al. "DDGun: An Untrained Method for the Prediction of Protein Stability Changes upon Single and Multiple Point Variations". In: *BMC Bioinformatics* 20.14 (July 2019), p. 335. URL: https://doi.org/10.1186/s12859-019-2923-1 (visited on 02/10/2022).

[26] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5998–6008. URL: https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 11/06/2020).

[27] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423 (visited on 04/21/2021).

[28] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018. URL: https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035 (visited on 03/01/2023).

[29] Alec Radford et al. "Language Models Are Unsupervised Multitask Learners". In: (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[30] Tom B. Brown et al. "Language Models Are Few-Shot Learners". In: *arXiv:2005.14165 [Cs]*. July 2020. URL: http://arxiv.org/abs/2005.14165 (visited on 11/30/2021).

[31] Richard Sutton and Andrew Barto. *Reinforcement Learning - An Introduction*. Second. MIT Press, 2018.

[32] Yaser Keneshloo et al. "Deep Reinforcement Learning for Sequence-to-Sequence Models". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.7 (July 2020), pp. 2469–2489.

[33] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8.3 (May 1992), pp. 229–256. URL: https://doi.org/10.1007/BF00992696 (visited on 03/03/2023).

S Supplementary Material

S.1 Variational Autoencoders

The basic idea behind AEs and VAEs is, that there are some hidden "latent" variables that can be used to reconstruct high dimensional data from a lower dimensional latent representation. For pictures of faces, these latent variables could be the length, color and shape of hair, the facial expression, etc.

AEs consist of an encoder that maps an input example x to a latent representation vector z and a decoder that takes in this vector and reconstructs the example. They are trained together to minimize the reconstruction error. Unfortunately, AE may suffer from "holes" in the latent space where there are no corresponding realistic examples. VAEs aim to address this by incorporating an additional KL-divergence term into their objective function, that encourages the latent representations to be normally distributed around the origin in latent space.

VAE may then suffer from another problem known as "posterior-collapse" or "KL-vanishing". This arises when using a very powerful decoder - often a AR model. Here the following strategy often ends up being optimal. The encoder simply learns to minimize the KL-divergence term while the decoder ignores the latent representations and directly learns to construct realistic examples. Various methods have been developed to mitigate it and preserve the meaningfulness of the latent representations. We follow an approach suggested by [17] which combines three methods.

- **Free bits:** The KL divergence term in the objective function is cut off by a hyper-parameter λ in each latent dimension. This allows for a controlled amount of information about the input to be retained in the latent variable without penalty.
- **Annealing:** During training, the KL-divergence term in the objective function gets weighted by a factor β. This starts out at zero - effectively allowing AE training for a certain period of time - before slowly rising to one.
- **Resetting the Decoder:** The uninformative latent representations provided by the encoder initially during training may lead to the decoder learning to disregard them. Therefore, when the AE training ends, the decoder is reset to its untrained state. This allows the decoder to learn from more meaningful latents provided by the encoder.

S.2 Transformers

Transformer networks [26] operate on a set of elements, each represented by a vector of dimensionality m, which undergoes transformation by multiple transformer layers. To incorporate positional information and convert tokens into vectors, each token needs to undergo an embedding process, mapping it to a m-dimensional space using an embedding layer. The attention mechanism allows elements to selectively focus on and incorporate relevant information from other elements, facilitating effective information exchange and processing within the transformer architecture.

The two main categories of transformer networks were already outlined in the original transformer architecture [26], that described a translation system consisting of an encoder- and a decoder. The encoder evolved into the Bidirectional Encoder Representations for Transformers (BERT) architecture [27], while the decoder inspired the Generative Pretrained Transformer (GPT) family of auto-regressive models [28, 29, 30], including ChatGPT.

- **Encoder:** When translating from for example English to French, the encoder stack would take in the English text sequence and convert it into a sequence of contextual embeddings of the same length - only attending to all English tokens during the process.
- **Decoder:** In contrast, the decoder would predicting each French word based on all previous French words in the sentence and the encoding provided by the encoder for the whole English text. Therefore, two modifications to the attention mechanism are required.

  Firstly, the decoder is an AR model, meaning it predicts the next token in a sequence based on the preceding tokens. To train AR models efficiently, teacher forcing is commonly used. This means that the entire target sequence, like the French translation, is provided at once. To prevent the model from attending to future information during training, a masking mechanism is employed, setting the weights of future values to zero.

  Secondly, in translation tasks, the decoder needs to have access to the encoding of the original input sequence, such as the English sentence encoded by the encoder. This is achieved through cross-attention. This allows the decoder to attend to and incorporate information from a different sequence. In contrast, self-attention - which is also used in the encoder - refers to attending only within the same sequence.

S.3 Reinforcement Learning

We want to backpropagate gradient information from the MHC-I presentation predictor, through the `CAPE-XVAE` decoder, to the latent space. However, directly backpropagating through the discrete decoder output is not possible, so we require an alternative method. One such approach is RL, which involves learning how to take actions to maximize a reward signal [31]. RL has been utilized for training the model parameters of sequence-to-sequence models [32]. In contrast, we employ RL to train the latent vector z, which steers the decoders behaviour.

The typical RL setting involves an agent observing its current state s_t and based on this taking an action a_t in an environment. The agent follows a policy π_θ, parameterized by θ, to select an action, which can be stochastic. After

performing the action, the environment provides the agent with a reward signal r_t based on the immediate benefit of the action in the environment. An episode consists of sequential iterations of this process until a terminal state is reached. The complexity of RL arises from the fact that actions taken at earlier stages of an episode can have repercussions on future rewards, either increasing or decreasing them. Consequently, the agent's objective is to maximize the (sometimes discounted) expected sum of rewards ($J(\theta) = \mathbb{E}_{\pi(\theta)} \sum_t r_t$) over the course of the episode. Additionally, we define the state value function $q_\pi(s, a)$, which represents the expected value of discounted rewards when taking action a in state s and subsequently following policy π. The state-value function serves as a useful estimate for evaluating the long-term desirability of actions within a given state.

The policy function $\pi_\theta(s_t)$ is crucial for maximizing the expected reward J. To find its optimal parameters θ (in our case the latent vector), the gradient ascent method shown in Equation 1 can be used:

$$\theta_{j+1} = \theta_j + \alpha \nabla J(\theta_j) \tag{1}$$

Directly backpropagating each trajectory through the discrete tokens is not possible. Fortunately, the REINFORCE [33] algorithm allows us to approximate $\nabla J(\theta_j)$ by the following Equation 2.

$$\nabla J(\theta_j) \approx \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right] \tag{2}$$

When we then define $\mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t), G_t = \sum_{k=0}^{\inf} R_{t+1+k}$ [31, Section 13.3], we get the following gradient ascent formula:

$$\theta_{j+1} = \theta_j + \alpha G_t \nabla \ln \pi_\theta(A_t|S_t)$$

All of the terms therein can be calculated by us.

S.4 Hyper-parameter Search

In a first step we ran 51 trials of a hyper-parameter search across several dimensions to minimize the modified VAE loss on the validation set (see Subsection 2.3.2). We arrived at the following best hyper-parameter configuration in the column **Selected**.

Hyper-parameter	**Distribution**	**Selected**
Number of latent variables (L)	1	1
Dimension of latent variables (l)	32	32
Lambda (λ)	1.0	1.0
Model dimensions (d)	256	256
Feed-forward dimensions	512	512
Number of heads	8	8
Number of layers	1 to 5	3
Dropout	Uniform(0, 0.4)	0.015
Learning rate	Log-Uniform(0.0001, 0.01)	0.00013
Batch size	32	32

Table 2: Hyper-parameter search

Each of the trials was trained for 500 epochs on an auto-encoder objective ($\beta = 0$). Then, the decoder weights were reset to their initial state. Over the next 250 epochs of training the β parameter would linearly be increase to reach one. Afterwards training ran for another 750 epochs with $\beta = 1$. Figure 6 shows the development of the train and validation losses during the training epochs. You can clearly see a spike when the CAPE-decoder got reinitialized. Also, we see some divergence between training and validation loss - pointing to some over-fitting. This might be caused by the relatively little data we used.

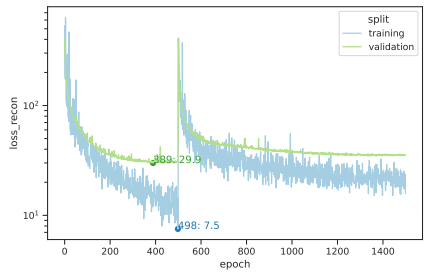


Figure 6: Reconstruction loss during training