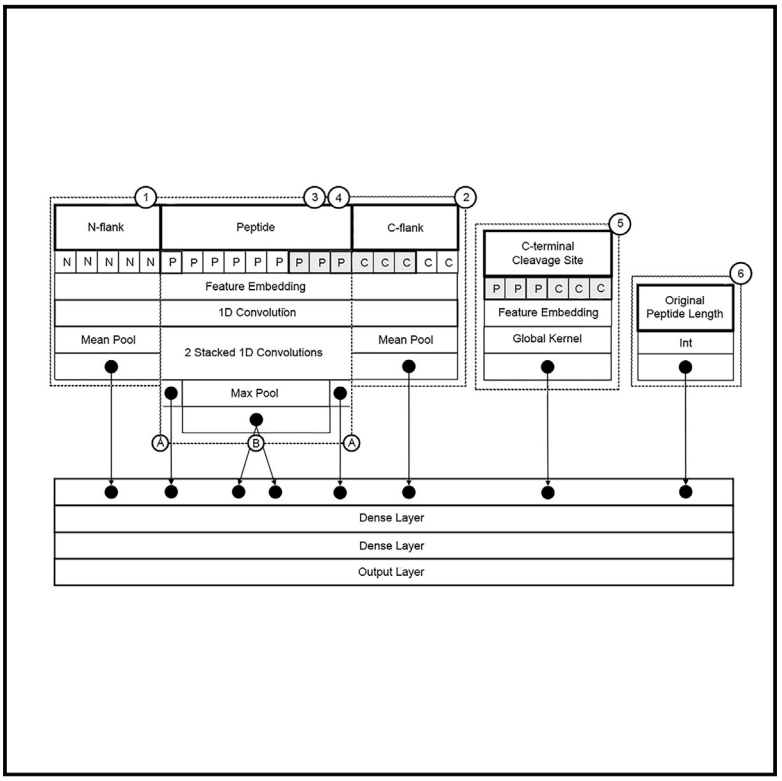
Cell Reports
Methods

Report

Improving MHC class I antigen-processing predictions using representation learning and cleavage site-specific kernels

Graphical abstract



Authors

Patrick J. Lawrence, Xia Ning

Correspondence

ning.104@osu.edu

In brief

Lawrence et al. develop a deep-learning model, MHCrank, to predict the probability of antigen processing for recognition by CD8$^+$ T cells. MHCrank outperforms existing antigen-processing predictors. Additionally, the model is open source, making it readily available for use in drug and vaccine development.

Highlights

- MHCrank, a new MHC class I processing predictor, outperforms existing methods
- Learned embeddings correlate with important properties for antigen processing
- Cleavage site-specific kernels identify relevant enrichment patterns for amino acids

Lawrence & Ning, 2022, Cell Reports Methods 2, 100293
September 19, 2022
https://doi.org/10.1016/j.crmeth.2022.100293

CellPress

Cell Reports Methods

CellPress
OPEN ACCESS

Report

Improving MHC class I antigen-processing predictions using representation learning and cleavage site-specific kernels

Patrick J. Lawrence[1] and Xia Ning[1,2,3,4,*]
[1]Biomedical Informatics Department, The Ohio State University, 1800 Cannon Drive, Lincoln Tower 250, Columbus, OH 43210, USA
[2]Computer Science and Engineering Department, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210, USA
[3]Translational Data Analytics Institute, The Ohio State University, 1760 Neil Avenue, Columbus, OH 43210, USA
[4]Lead contact
*Correspondence: ning.104@osu.edu
https://doi.org/10.1016/j.crmeth.2022.100293

MOTIVATION More than binding affinity, designing effective drugs and vaccines to stimulate the adaptive immune response requires accurate predictions regarding which antigens will be produced through processing and ultimately presented by MHC class I molecules. Constructing a model that is more acutely aware of biologically relevant features for processing and presentation will improve these predictions. However, future predictions will benefit from considering amino acid structure in addition to the antigen sequence.

SUMMARY

In this work, we propose a new deep-learning model, MHCrank, to predict the probability that a peptide will be processed for presentation by MHC class I molecules. We find that the performance of our model is significantly higher than that of two previously published baseline methods: MHCflurry and netMHCpan. This improvement arises from utilizing both cleavage site-specific kernels and learned embeddings for amino acids. By visualizing site-specific amino acid enrichment patterns, we observe that MHCrank's top-ranked peptides exhibit enrichments at biologically relevant positions and are consistent with previous work. Furthermore, the cosine similarity matrix derived from MHCrank's learned embeddings for amino acids correlates highly with physiochemical properties that have been experimentally demonstrated to be instrumental in determining a peptide's favorability for processing. Altogether, the results reported in this work indicate that MHCrank demonstrates strong performance compared with existing methods and could have vast applicability in aiding drug and vaccine development.

INTRODUCTION

The major histocompatibility complex (MHC) class I protein is a vital part of the immune system's response to intracellular invasion by viruses, bacteria, and parasites and against tumorigenesis (Comber and Philip, 2014). Its primary responsibility is to present antigens—short peptides 8–10 amino acids in length that are cleaved from proteins—into the extracellular environment to be recognized by cytotoxic ($CD8^+$) T cells, which subsequently eliminate compromised cells via apoptosis (Rock et al., 2004). Thus, these peptides can be leveraged for the development of both vaccines that prime $CD8^+$ T cells against a pathogen and drugs that elicit cytolytic activity in tumor cells.

There is not a single MHC class I molecule. Rather, multiple versions can be produced based on the human leukocyte antigen (HLA) alleles present in an individual's genome. HLA is the portion of the MHC class I molecule that binds presented peptides; hypermutability within the HLA binding groove yields variability in the binding affinity of processed peptides and affords greater coverage of the number of pathogens that can be recognized (Wieczorek et al., 2017).

The peptides presented by MHC class I molecules must first undergo a series of processing steps to make the peptide more favorable for presentation. Peptidases digest proteins into fragments based on identified consensus sequences that indicate where to cleave proteins (Rock and Goldberg, 1999). Fragments from digested protein are then translocated across the rough endoplasmic reticulum (RER) membrane by the TAP protein (Kloetzel, 2001; Lundegaard et al., 2010). TAP filters these peptides, based on which ones are most likely to have high affinity for the MHC class I molecule. Specifically, TAP has a higher affinity for peptides between 8 and 16 amino acids

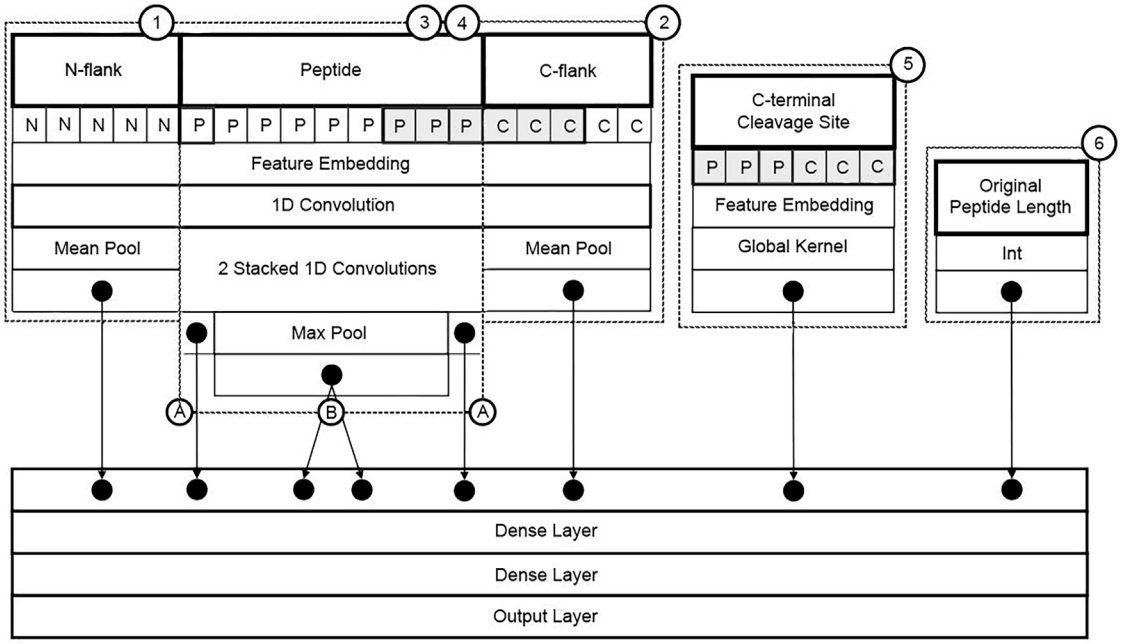


Figure 1. MHCrank model architecture

MHCrank takes a uniform-length N-flank + peptide + C-flank sequence, C-terminal cleavage site (see gray box), and the peptide's original length before padding or trimming as input. The amino acids comprising the sequence and cleavage site-specific kernel (CSSK) undergo feature embedding. A convolution layer is applied to the embedding of the entire sequence. The remainder of the MHCrank architecture can be split into six components. Component (1) applies a mean pool to the convolution output corresponding to the N-flank. Component (2) applies a mean pool to the convolution output corresponding to the C-flank. The convolution output corresponding to the peptide sequence is forwarded to two stacked convolution layers. Components (3) and (4) each have two outputs (A and B) obtained from the output of these convolution layers. (3A) extracts the output corresponding to the peptide's N-terminal amino acid. (4A) extracts the output corresponding to the peptide's C-terminal amino acid. (3B) applies a mean pool to the peptide's non-N-terminal amino acids. (4B) applies a mean pool to the peptide's non-C-terminal amino acids. Component (5) applies a global kernel to the embedded CSSK. Component (6) is a single node that takes the peptide's original length as input. Two dense layers are applied to the concatenated output of each component. The output from the second dense layer enters an output layer that predicts the probability of the input peptide undergoing antigen processing. Note that the layout of this diagram is largely inspired by the presentation of MHCflurry's architecture (O'Donnell et al., 2020).

See also Figure S1.

in length (Abele and Tampé, 2004), as well as peptides with either hydrophobic or basic C-terminal amino acids (Kloetzel, 2001; Rock et al., 2004). Once in the RER, longer peptides may be further cleaved from the N termini to optimize its binding affinity (Rock and Goldberg, 1999), but the C terminus remains untouched as this is the primary anchor point between the antigen and MHC molecule. Thus, leveraging these peptides for vaccine and drug development requires an understanding of which peptides will have the greatest opportunity to bind to any given HLA allele.

As a result, computer-aided methods have been developed to identify candidate peptides (O'Donnell et al., 2020; Jurtz et al., 2017; Comber and Philip, 2014; Phloyphisut et al., 2019; Boehm et al., 2019; Zeng and Gifford, 2019). Among them, deep-learning (DL) models that rank peptides' binding affinities to MHC class I molecule(s) have achieved superior performance (Jurtz et al., 2017). These models are created with the goal of predicting which peptides will have the highest binding affinity for the HLA alleles. However, there is no guarantee that highly ranked peptides will be selected for presentation by upstream proteins (O'Donnell et al., 2020), meaning that these models may lack biological relevance. Recent attempts have been made to develop models that rank the likelihood of peptides being processed within the MHC class I presentation pathway (O'Donnell et al., 2020; Jurtz et al., 2017). These are considered HLA independent as they do not require any information about the HLA alleles, making them a more generalized approach. By training on peptides that have been confirmed to be processed for and presented by MHC class I molecules, in combination with the amino acid residues immediately flanking the peptides in their original protein, such models can—in theory—learn the features that make a peptide more likely to be cleaved and processed for presentation. This incorporates the biological information missing from binding affinity models and has the potential to enable superior performance on predicting presentation.

In this work, we propose a novel DL, antigen-processing (AP) prediction model, denoted MHCrank, that has been developed to rank candidate peptides by their likelihood to be processed for MHC class I presentation. The architecture for MHCrank is presented in Figure 1. Further details can be found in STAR Methods. Based on the architecture used by O'Donnell et al. (2020), our model imparts additional biological relevance, focusing on the carboxyl (C)-terminal cleavage site of the antigen and pre-processing antigen sequences to simulate what is

CellPress
OPEN ACCESS

Table 1. Performance comparison: Mean AUC

MHCrank ensembles						
EL	Fw-top1	Fw-top2	Ba-top1	Ba-top2	C-top1	C-top2
0.9050	0.9073† (0.0)****	0.9120† (0.0)****	0.9102† (0.0)****	0.9147† (0.0)****	0.9121† (0.0)****	0.9153† (0.0)****

Results comparing the performance of MHCrank's ensembles against EL (netMHCpan4.0-EL) with respect to mean AUC. † highlights MHCrank ensemble's improvement in performance over netMHCpan4.0-EL; p values are reported in parentheses to the right of the mean performance values. Statistically significant improvement of MHCrank's ensembles relative to netMHCpan4.0-EL's performance (after multiple-hypothesis correction) is denoted as follows: ****$p \leq 0.001$. See also Figure S4 and Tables S2–S4.

observed *in vivo* (O'Donnell et al., 2020). In our development of MHCrank, we forgo the use of the widely used BLOSUM62 matrix for amino acid representations. Instead, MHCrank learns a problem-specific embedding for each amino acid. Our experiments on the benchmark dataset demonstrate that MHCrank achieved a significant performance improvement over the compared baseline methods: netMHCpan-4.0 eluted ligand and MHCflurry-2.0 antigen processing, denoted netMHCpan4.0-EL and MHCflurry-AP, respectively.

Our paper is organized as follows. We first trained and evaluated the proposed MHCrank model using data published by O'Donnell et al. (2020). Further discussion regarding both the training and testing data sets can be found in the "training data" and "testing data processing" subsections of STAR Methods. We evaluated our model by comparing its performance against the MHCflurry-AP and netMHCpan4.0-EL baselines using three metrics: area under the curve (AUC), precision@*k*, and NDCG@*k*. Detailed descriptions of these metrics can be found in the "evaluation metrics" subsection of STAR Methods. Finally, we added transparency to our model by identifying what it had learned. We achieved this both by analyzing the enrichment patterns of top-ranked peptides from our model and by comparing the amino acid embeddings that were learned by our model.

RESULTS

Performance evaluation

The validation AUC measured during training across four folds (Table S1) was used to select models to be included in six MHCrank ensembles: Fw-top1—the combination of the best-performing model trained on each fold; Fw-top2—the combination of the top two performing models trained on each fold; Ba-top1—the combination of the models from the same hyperparameter set with the highest average validation AUC; Ba-top2—the combination of the models from the two hyperparameter sets with the highest average validation AUC; C-top1—the combination of the Fw-top1 and the Ba-top1 ensembles; and C-top2—the combination of the Fw-top2 and the Ba-top2 ensembles. A more comprehensive description of the ensembles is presented in the "ensemble methods and model selection" subsection of STAR Methods. All (100%) of the models selected for inclusion in the ensembles utilized the embedding amino acid representation method (Table S7A). Examining Table S1, we observe a decrease in AUC from training to validation for all models. This is expected, as models will have learned information from the peptides in the training set, while the peptides in the validation set can be considered novel. Interestingly, we observed larger drops in AUC from the training to the validation set for the two best-performing models of those trained on fold 3 compared with the top-performing models from other folds. Furthermore, the best-performing model of those trained on fold 3 exhibits a large standard deviation in both mean training and validation AUC compared with all other selected models. This is because the model with this set of hyperparameters achieved poor performance when trained on fold 0. This suggests that the hyperparameters used by this model are not conducive to learning the important features of fold 0's data. Moreover, the only hyperparameter that is shared by the best-performing models from fold 3 and not present in any of the other top models is that those from fold 3 use an input peptide length of 15. This indicates that perhaps a larger proportion of peptides in fold 3 have central amino acids that contribute to MHC class I processing or binding than those of the other folds.

Tables 1, 2, and 3 and Tables S2A–S2C compare the performance of all six MHCrank ensembles against netMHCpan4.0-EL and MHCflurry-AP, in terms of mean AUC, precision@*k*, and NDCG@*k*, respectively. Note that the significance levels ($p \leq 0.1, 0.05, 0.01, 0.001$) denoted in Tables 1, 2, 3, and S2 are reported following multiple-hypothesis correction. We find that all of the MHCrank ensembles perform significantly ($p \leq 0.001$) better than MHCflurry-AP for all evaluated metrics with the exception of Fw-top1's AUC and Ba-top1's AUC and precision@*k* and NDCG@*k* for $k = 10$ (Table S2).

Evaluation of AUC

Tables 1, 2, 3, and S2A show that, for all the methods, the mean AUC was higher than 0.9. This suggests all methods have learned to distinguish between hits and decoys. Among the evaluated methods, C-top2 achieves the best performance (0.9153). All the MHCrank ensembles outperform netMHCpan4.0-EL (Table S2A), and four of the six MHCrank ensembles—Fw-top2, Ba-top2, C-top1, and C-top2—outperform MHCflurry-AP; each achieves statistically significant improvement at the level of $p \leq 0.001$ (Table 1). This demonstrates the strong power of MHCrank ensembles in learning from the training data to score hits above decoys. Further discussion about AUC is available in the section "evaluating training and testing hits and decoys." Additionally, we evaluate the impact of the ratio of decoys-to-hits on model performance with respect to AUC and precision@*k* and NDCG@*k* in the section "evaluation of model performance with reduced decoy-to-hit ratios."

Evaluation of precision@k

Table S2B illustrates that all MHCrank ensembles consistently outperform MHCflurry-AP across all *k* values with statistical significance. Furthermore, four of the six MHCrank ensembles (Fw-top1, Fw-top2, C-top1, and C-top2) outperform netMHCpan4.0-EL for small values of *k* (10, 25) (see Table 2). For $k = 500$, all

Table 2. Performance comparison: Mean precision@*k*

MHCrank							
k	EL	Fw-top1	Fw-top2	Ba-top1	Ba-top2	C-top1	C-top2
10	0.7065	0.7260† (0.0028)**	0.7251† (0.0050)**	0.6492 (0.0)	0.6980 (0.1995)	0.7089† (0.7117)	0.7116† (0.4440)
25	0.6434	0.6544† (0.0114)*	0.6517† (0.0524)	0.6314 (0.0047)	0.6246 (0.0)	0.6496† (0.1424)	0.6452† (0.6729)
50	0.5971	0.5973† (0.9488)	0.5971 (1.0)	0.5891 (0.0105)	0.5922 (0.1142)	0.6063† (0.0031)**	0.6035† (0.0391)
100	0.5495	0.5284 (0.0)	0.5360 (0.0)	0.5182 (0.0)	0.5304 (0.0)	0.5491 (0.8511)	0.5434 (0.0074)
250	0.4344	0.4202 (0.0)	0.4410† (0.0)****	0.4198 (0.0)	0.4359† (0.3132)	0.4412† (0.0)****	0.4482† (0.0)****
500	0.3288	0.3413† (0.0)****	0.3544† (0.0)****	0.3409† (0.0)****	0.3552† (0.0)****	0.3550† (0.0)****	0.3604† (0.0)****

Results comparing the performance of MHCrank's ensembles against EL (netMHCpan4.0-EL) with respect to mean precision@*k*. † highlights MHCrank ensemble's improvement in performance over netMHCpan4.0-EL; p values are reported in parentheses to the right of the mean performance values. Statistically significant improvement of MHCrank's ensembles relative to netMHCpan4.0-EL's performance (after multiple-hypothesis correction) is denoted as follows: *$p \leq 0.1$; **$p \leq 0.05$; ****$p \leq 0.001$. See also Figure S4 and Tables S2–S4.

MHCrank ensembles outperform netMHCpan4.0-EL with a significance of at least $p \leq 0.001$. Among the six MHCrank ensemble methods, Ba-top1 and Ba-top2 are generally the worst-performing ensembles in terms of precision@*k*. On average, neither ensemble improves upon the performance of netMHCpan4.0-EL. Furthermore, Ba-top1 is the only ensemble to achieve a significantly reduced precision@*k* ($p \leq 0.1$) relative to MHCflurry-AP for any value of *k* (*k* =10). These two ensemble methods used the overall best hyperparameters across all the four folds. Thus, the models trained on each fold using these hyperparameter sets were not necessarily optimized for that fold. Consequently, the combination of these suboptimal models did not produce the best performance. On the contrary, Fw-top2 and C-top1 were the two ensemble methods that achieved the overall best precision@*k*. Both methods incorporated at least the best model (C-top1) or two best models (Fw-top2) for each of the four folds, allowing the ensemble to integrate the most predictive power possible from the data.

Evaluation of NDCG@k

Tables 3 and S2C display very similar trends to those in Tables 2 and S2B. That is, all the MHCrank methods outperform MHCflurry-AP (Table S2C), and four of the six ensemble methods (Fw-top1, Fw-top2, C-top1, and C-top2) outperform netMHCpan4.0-EL (Table 3). Again, Ba-top1 and Ba-top2 are the worst-performing ensemble methods. Unlike Table 2, in Table 3 we observe that Fw-top1 and Fw-top2 are the two best-performing ensembles, with each achieving significant improvements over netMHCpan4.0-EL for three and four values of *k*, respectively. This indicates that Fw-top2 is able to rank more hits at higher positions in the ranking. C-top1 and C-top2 combine Fw-top1 and Ba-top1, and Fw-top2 and Ba-top2, respectively. This grants C-top1 and C-top2 both pros and cons of both types of ensembles. Thus it is intuitive that they, in consequence, achieved mid-level performance.

Percent improvement

Tables 4 and S2D summarize the improvement in performance of MHCrank as a percentage change of its best-performing ensemble—Fw-top2—relative to the performance of the netMHCpan4.0-EL and MHCflurry-AP baselines, respectively, for mean AUC, precision@*k*, and NDCG@*k*. For both precision@*k* and NDCG@*k*, the percent improvement garnered by Fw-top2 over both netMHCpan4.0-EL and MHCflurry-AP exhibits generally increasing performance with increasing values of *k* (e.g., *k* = 10 versus *k* = 500). All percent improvements over MHCflurry-AP are significant for both metrics ($p \leq 0.001$). This again indicates that Fw-top2 is able to more effectively rank peptides likely to be processed for presentation among the very top of ranking lists when compared with netMHCpan and MHCflurry. One surprising result is the dip in both precision@*k* and NDCG@*k* of Fw-top2 relative to netMHCpan4.0-EL for *k* = 50 and 100, followed by rapid improvement between *k* = 250 and 500. One plausible explanation is netMHCpan4.0-EL learned to highly rank peptides with a specific motif that is highly enriched within hits (Figure 2). After exhausting all peptides containing the learned motif, the accuracy of subsequently ranked peptides would likely deteriorate.

Evaluation of model performance with reduced decoy-to-hit ratios

The specific ratio of 99-decoys-per-hit that we use for evaluating model performance is a convention utilized by many the state-of-the-art methods: MHCflurry (O'Donnell et al., 2020), DeepLigand (Zeng and Gifford, 2019), and netMHCpan (Jurtz et al., 2017). The reason for this is that the engineered class rarity is designed to replicate biological conditions. Specifically, a very small subset of produced peptides is selected by the TAP protein to be transported into the ER for further processing and to ultimately be presented (Yewdell et al., 2003).

To further understand the role that the data's composition plays in the performance of each model, we recalculated AUC, precision@*k*, and NDCG@*k* with 50 decoys per hit and 20 decoys per hit. The set of hits in the amended test data is identical to that present in the original test data. The set of decoys in each of the amended test sets was randomly sampled from the set of decoys in the original test data. We report the results of these experiments in Table S3.

Tables S3B and S3A show that the ratio of decoys per hit (50 and 20, respectively) does not affect a model's AUC. The performance of all models evaluated using 50 decoys per hit and 20 decoys per hit was nearly identical to that reported in Table 1. However, both the precision@*k* (Table S3D) and NDCG@*k* (Table S3F) were improved both for each model and for each value of *k* when using 50 decoys per hit. The improvement of all models with a reduced decoy-per-hit ratio indicates that the difficulty of the task has decreased, as there are fewer decoys available to be ranked above hits. Interestingly, 50 decoys per

Table 3. Performance comparison: Mean NDCG@*k*

MHCrank ensembles							
K	EL	Fw-top1	Fw-top2	Ba-top1	Ba-top2	C-top1	C-top2
10	0.7253	0.7451† (0.0032)**	0.7544† (0.0)****	0.6705 (0.0)	0.7166 (0.2004)	0.7265† (0.8571)	0.7357† (0.1268)
25	0.6712	0.6846† (0.0031)**	0.6877† (0.0002)***	0.6486 (0.0)	0.6547 (0.0003)	0.6761† (0.2813)	0.6770† (0.1905)
50	0.6272	0.6317† (0.17375)	0.6344† (0.0282)	0.6113 (0.0)	0.6198 (0.02749)	0.6346† (0.02602)	0.6348† (0.0213)
100	0.5795	0.5658 (0.0)	0.5735 (0.01295)	0.5486 (0.0)	0.5621 (0.0)	0.5805† (0.0)	0.5770 (0.0)
250	0.4712	0.4202 (0.0)	0.4780† (0.0)****	0.4538 (0.0)	0.4701 (0.4678)	0.4772† (0.0001)****	0.4833† (0.0)****
500	0.3685	0.3779† (0.0)****	0.3909† (0.0)****	0.3743† (0.0)****	0.3891† (0.0)****	0.3909† (0.0)****	0.3961† (0.0)****

Results comparing the performance of MHCrank's ensembles against EL (netMHCpan4.0-EL) with respect to mean NDCG@*k*. † highlights MHCrank ensemble's improvement in performance over netMHCpan4.0-EL; p values are reported in parentheses to the right of the mean performance values. Statistically significant improvement of MHCrank's ensembles relative to netMHCpan4.0-EL's performance (after multiple-hypothesis correction) is denoted as follows: **$p \leq 0.05$; ***$p \leq 0.01$; ****$p \leq 0.001$. See also Figure S4 and Tables S2–S4.

hit yielded a larger improvement in MHCrank's performance relative to netMHCpan4.0-EL. Compared with Table 2, which used 99 decoys per hit, MHCrank's improvement in precision@*k* = 25 becomes significant. Additionally, compared with Table 3, MHCrank achieved significant improvement in NDCG@*k* over netMHCpan4.0-EL from *k* = 25 to *k* = 100. While the difficulty of the task was reduced, the larger relative improvement of MHCrank indicates that it is better at ranking hits above decoys and that there were fewer among the top values of *k* than in either MHCflurry-AP or netMHCpan4.0-EL.

Similar to the trend observed for 50 decoys per hit in Tables S3B, S3D, and S3F, the use of 20 decoys per hit (Tables S3A, S3C, and S3E) to evaluate model performance yields improvements over the use of both 50 decoys per hit and 99 decoys per hit. However, compared with the precision@*k* and NDCG@*k* reported for 50 decoys per hit (Tables S3D and S3F, respectively), the significance of MHCrank's improvements over netMHCpan4.0-EL are reduced when using 20 decoys per hit for both precision@*k* and NDCG@*k* (Tables S3C and S3E, respectively). This result suggests that the difficulty of the prediction tasks has been further reduced compared with using both 99 and 50 decoys per hit. Despite the triviality of the decoys relative to hits, these results favor the use of 99 decoys per hit. Not only is model evaluation using 99 decoys per hit standard practice, but this ratio also replicates biological conditions and introduces difficulty to the prediction tasks used for performance evaluation, both of which allow for better comparisons.

Enrichment of training and top-ranked peptides

Figures 2 and S2 show the position-specific enrichment in various sets of peptides. This is done to assess how peptide composition may impact both training and testing. The second and ninth positions are underscored in each figure with yellow boxes. These positions correspond to the canonical anchor residues for binding to the MHC class I molecule.

Evaluating training peptides

Figure 2A illustrates the position-specific enrichment for 50,000 randomly selected hits from the training data. In this figure, position 9 exhibits a high level of enrichment. The amino acids that are enriched at this position are either hydrophobic (e.g., L, V, and I) or aromatic (F, Y, W) and are all enriched to comparable levels. This conforms to the biological relevance affirmed by previous studies (Wieczorek et al., 2017) that hydrophobic residues tend to be favored in the C-terminal position. Position 2 also has slightly elevated enrichment when compared with the low levels of enrichment that exist for positions 1 and 3–8. Taken together, this demonstrates that the peptides selected for training can represent a large range of different peptides.

Evaluating training and testing hits and decoys

The enrichments reported in Figure S2 were obtained to determine the utility of AUC and precision@*k* metrics. This includes whether or not either could be considered a biased metric. In Figures S2A and S2C, we observed no noticeable difference in the enrichment patterns of training and testing hits. This is promising, as it suggests that both are representative processed antigens, making the data biologically relevant. Additionally, there is not a noticeable difference between training hits (Figure S2A) and decoys (Figure S2B). This indicates that the training decoys are non-trivial. The peptides used for training were selected from a larger dataset using scores from MHCflurry's BA model as criteria. As discussed by O'Donnell et al., the intuition behind using hits and decoys with the top 2% of predicted binding affinities (i.e., decoys are very similar to hits) was to prevent the model from learning features associated with binding affinities (O'Donnell et al., 2020). This effectively forces the AP model to learn non-trivial features related to antigen processing. However, unlike the training data, there is a substantial difference between the enrichment patterns of the hits and decoys in the testing data, as demonstrated in Figures S2C and S2D. This is because the testing data are from a different data source (O'Donnell et al., 2020) that was published after the training data. Unlike the training data, after randomly sampling decoys from the protein from which each peptide originates, the testing data are not filtered by binding affinity. Here, we want to point out that the formulation of the testing set, including true hits and sampled decoys, was done according to a process that is well accepted by the research community and widely used by the state-of-the-art methods, including MHCflurry, DeepLigand, and netMHCpan, to evaluate their performance. While the use of AUC as a metric is common practice, given the dissimilarity between testing hits and decoys, the triviality of testing decoys likely inflates this metric. Even so, such inflation would likely affect all evaluated models to a comparable extent. As such, the relative ordering of AUC values may still be considered a useful metric and may provide valuable insights.

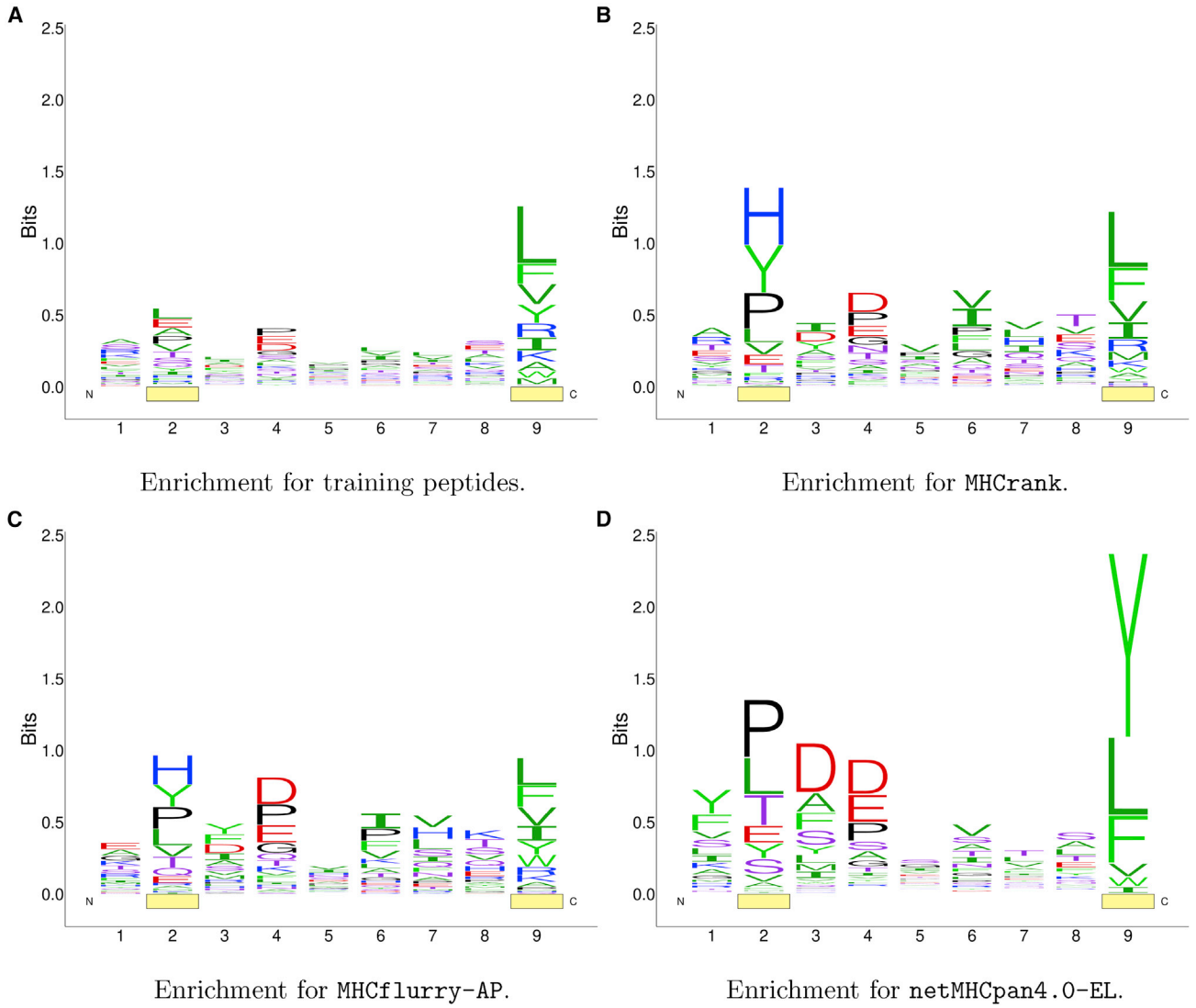


Figure 2. Amino acid enrichment of training peptides and top 100 predicted candidates

(A–D) The enrichment of amino acids in (A) 50,000 randomly sampled hits from training dataset and in the top 100 peptides from the testing data ranked by (B) MHCrank's Fw-top2 ensemble and both the (C) MHCflurry-AP and (D) netMHCpan4.0-EL baseline methods. Yellow boxes covering positions 2 and 9 in each figure highlight the enrichment of the peptides at their typical anchor positions.

See also Figures S2 and S3.

Evaluation of MHCrank's top-ranked peptides

Figure 2B depicts the position-specific enrichment for the set of the top 100 ranked peptides by the best MHCrank ensemble Fw-top2. The top 100 peptides were selected for each model as k = 100 is the threshold where MHCrank's performance rapidly improves compared with netMHCpan4.0-EL. By examining the general composition of peptides ranked highly by MHCrank and the netMHCpan4.0-EL and MHCflurry-AP baselines, we believe we may be able to elucidate the reason for the drastic shift in performance at k = 100 that we observed in Table 4. The position-specific enrichment of Fw-top2's recommended peptides shows an enhancement in the enrichment at the second and ninth positions relative to Figure 2A. Their enhancement in the Fw-top2 ensemble's top 100 ranked peptides indicates that our method is capable of discerning both anchor positions within the peptides. Moreover, similar to the enrichment of the training peptides, the patent enrichment at the ninth position features uniform enrichment of mostly hydrophobic amino acids. This suggests that Fw-top2 learned to identify the features and physiochemical properties of the residues versus specific sequences. Note there was also a slight increase in the enrichment of the central amino acids (positions 3–9) compared with Figure 2A, indicating that MHCrank may have learned some motifs that convey processing favorability within the central amino acids.

Evaluation of MHCflurry-AP's top-ranked peptides

Figure 2C depicts the position-specific enrichment for the set of top 100 ranked peptides by MHCflurry-AP. Like MHCrank,

Table 4. Performance comparison: Percent change

k	Precision@*k*	NDCG@*k*	AUC
10	2.63†**	4.00†****	0.77†****
25	1.29†	2.47†***	–
50	0.00	1.15†	–
100	−2.47	−1.05	–
250	1.51†****	1.45†****	–
500	7.77†****	6.09†****	–

Results comparing the performance of MHCrank's ensembles against EL (netMHCpan4.0-EL) with respect to percent improvement. †highlights MHCrank ensemble's improvement in performance over netMHCpan4.0-EL. Statistically significant improvement of MHCrank's ensembles relative to netMHCpan4.0-EL's performance (after multiple-hypothesis correction) is denoted as follows: **$p \leq 0.05$; ***$p \leq 0.01$; ****$p \leq 0.001$. The p values are not reported for percent improvements, as these are derived from the performance reported for Fw-top2 (Tables 1, 2, and 3). Note that the reported percent improvement in AUC was calculated over the entire dataset, not at a specific *k* threshold. See also Figure S4 and Tables S2–S4.

MHCflurry-AP exhibited enhanced enrichment of the second position. However, this is overshadowed by similar enrichment levels of its central amino acids (positions 3–9). Additionally, Figure 2C displays a reduction in enrichment for the vital C-terminal position. Thus, it appears there was not any position for which MHCflurry-AP was able to learn meaningful features or trends. This lack of fit might explain why MHCflurry-AP's performance was worse than MHCrank's performance.

Evaluation of netMHCpan4.0-EL's top-ranked peptides

Figure 2D depicts the position-specific enrichment for the set of the top 100 ranked peptides by netMHCpan4.0-EL. The enrichment of the ninth position is higher than any position-specific enrichment from both MHCrank and MHCflurry-AP. Unlike the enrichment at this position present in MHCrank's top peptides, only three amino acids (Y, L, F) are enriched for netMHCpan4.0-EL. This is an important distinction because rather than learning properties of the amino acids that occupy this position in hits, it is likely that netMHCpan4.0-EL learned to prefer peptides that ended in one of the three enriched amino acids. In fact, 91% of netMHCpan4.0-EL's top 100 peptides feature either Y, L, or F in the C-terminal position, suggesting that netMHCpan4.0-EL's performance may decline when testing it with peptides that do not match this pattern. This adds credence to our explanation underlying the dip in both precision@*k* and NDCG@*k* of Fw-top2 relative to netMHCpan4.0-EL for $k = 50, 100$, followed by rapid improvement between $k = 250, 500$ (Table 4). Also noteworthy is netMHCpan4.0-EL's enhanced enrichment at positions 2, 3, and 4. The enrichment of positions 2–4 suggests that netMHCpan4.0-EL was able to learn that there is an important feature near those positions, but not which position was most informative.

Evaluation of allelic biases in site-specific amino acid enrichment and MHCrank's allele-specific performance

Even though MHCrank is HLA allele agnostic, we sought to ascertain whether there were any allelic biases in the peptides presented. This is important for vaccine and drug development, as it can determine whether a candidate peptide may be less effective among individuals in the population without the alleles favoring the motif(s) present in a given peptide. We obtained a monoallelic dataset published alongside MHCflurry (O'Donnell et al., 2020) (datafile: *DataS2*). We retained only hits from HLA-peptide combinations that were not present in our training data. We then obtained the site-specific amino acid enrichment of hits that were bound to each of the 92 HLA alleles present in the dataset. By studying the enrichment patterns of these alleles, we aimed to identify some of the features of peptides favored by these alleles. Doing so would facilitate an evaluation of MHCrank's allele-specific performance.

In Figure S3A, we present the enrichment of all hits in the processed dataset. The enrichment pattern observed here is consistent with the enrichment reported for both the training and testing hits (Figures S2A and S2C, respectively). Figures S3B–S3D are presented to provide representative examples of the three most common enrichment patterns observed within the dataset. Prevalence was determined by the number of hits from alleles with these enrichment patterns. The enrichment pattern of HLA*A:02-01, depicted in Figure S3B, represents the alleles that favor peptides with hydrophobic amino acids present in the anchor residues and/or the peptide's termini. Roughly 70% of the hits in the dataset were from interactions between peptides and alleles in this category.

Figure S3C presents the enrichment pattern of HLA*A:30:01. Unlike the enrichment pattern in Figure S3B, there is a noticeable decrease in the enrichment of hydrophobic amino acids. Instead, there is enrichment of basic residues at both the N and C termini. This indicates that HLA*A:30:01 and similar alleles favor peptides with basic termini for presentation. However, note that the total enrichment patterns of this dataset (Figure S3A) and the enrichment patterns of the training hits (Figure S2A) exhibit low-level enrichment of basic amino acids at the C-terminal anchor residue. This would suggest that possessing basic residues in the C terminus is not an uncommon feature among peptides presented by MHC class I molecules. In fact, approximately 16% of the hits in this dataset were interactions between peptides and alleles in this category.

Finally, Figure S3D presents the enrichment of HLA*A:36-01. This enrichment pattern exhibits high enrichment of acidic amino acids near the N terminus of presented peptides. Given the extremely low enrichment of acidic amino acids at any site for all hits in the training, testing, and monoallelic data sets (Figures S3D, S2A, and S2C, respectively) It may be inferred that alleles that prefer peptides with an acidic N terminus are less common than the previous two categories. Indeed, fewer than 10% of hits in this dataset are from alleles possessing a similar enrichment pattern to HLA*A:36-01.

We also obtained predictions for all the hits in the dataset. Using the average score for hits of alleles with enrichment profiles similar to Figure S3B as a baseline, we calculated the average change in predicted score both for hits belonging to alleles with an enrichment pattern similar to Figure S3C and for hits belonging to alleles with an enrichment pattern similar to Figure S3D. We found that hits from alleles with a basic enrichment profile garnered a 2.2% improvement. Additionally, the hits from alleles with an acidic enrichment profile had a significant reduction in average score at −16.7%. However, the reduced capacity at scoring acidic hits is likely the result of there being limited representative examples

within the training data. The similar scoring of basic hits relative to hydrophobic hits strengthens this claim.

Amino acid embedding: Learned versus hard-coded features

To ascertain whether the embedding method employed by MHCrank enhanced its performance and facilitated its observed improvement over the MHCflurry-AP and netMHCpan4.0-EL methods, we retrained all the models within the top-performing MHCrank ensemble (Fw-top2) six times. For each iteration, a distinct embedding method was utilized: embedding, em-BLO, BLOSUM, NormBLO, PC-NormBLO, and PC. Further details regarding these embedding methods can be found in "amino acid representation." Figure 3A–3C highlights the results of this experiment, specifically the average precision@*k*, NDCG@*k*, and AUC, respectively.

Evaluation of precision@k and NDCG@k

In terms of precision@*k*, MHCrank performance is significantly improved when using the embedding method than when using any other embedding method; this holds true for all values of *k* (Figure 3A). Models trained with the em-BLO embedding method performed the next best after the embedding method. Models trained using the PC embedding method performed significantly worse than when trained using any other embedding method. Interestingly, the concatenation of the normalized BLOSUM embedding with the PC embedding does not recapitulate the performance observed when using the normalized BLOSUM embedding alone. This suggests that expressiveness of the PC embedding is likely not the main issue underlying its poor performance. Rather, it is probable that some of the properties included, such as molecular weight, are not important features for predicting a peptide's favorability to be processed for MHC class I presentation. Furthermore, we observe the same trends when considering NDCG@*k* (Figure 3B).

Evaluation of AUC

The choice of embedding method follows a similar trend in AUC as well (Figure 3C). That is, the embedding, em-BLO, and BLOSUM embedding methods achieve the top performances. Additionally, the PC method again yields the worst performance of any embedding method. Note, however, that the AUC for MHCrank models trained with the embedding, em-BLO, and BLOSUM methods achieve similar performances. In fact, the AUC garnered through use of either the em-BLO or BLOSUM methods were not significantly different from the AUC achieved by using the embedding method. Thus, it may still be concluded that enabling MHCrank to learn amino-acid-specific embeddings contributed to its ability to identify peptides more likely to be processed.

Similarities of learned amino acid embeddings

To better understand what it is about the embedding method that enabled such substantial improvement in performance, we extracted the 21-dimension embedding vector for each amino acid from a representative MHCrank model. Figure 3D illustrates the cosine similarities among the embeddings learned by MHCrank for each amino acid. Amino acids in Figure 3D have been grouped according to their type: hydrophobic, aromatic, basic, acidic, polar, and other. We observed that, in general, the learned embeddings of amino acids within the same groups (i.e., of the same types) are more similar than those of amino acids from different groups (i.e., of different types). This indicates that MHCrank was capable of learning meaningful information from amino acids that may correlate with their physicochemical properties, and thus facilitate better predictions. This is further demonstrated by the similarities of learned embeddings of amino acids between certain groups. Figure 3D shows that the embeddings of aromatic and hydrophobic amino acids are more similar to each other than the other amino acid types. Likewise, the embeddings of basic, acidic, and polar amino acids are more similar to each other than they are to other amino acid types. This distinction suggests that MHCrank is capable of learning information that corresponds to an amino acid's hydrophilicity, an important physicochemical property involved in identifying peptides likely to be processed.

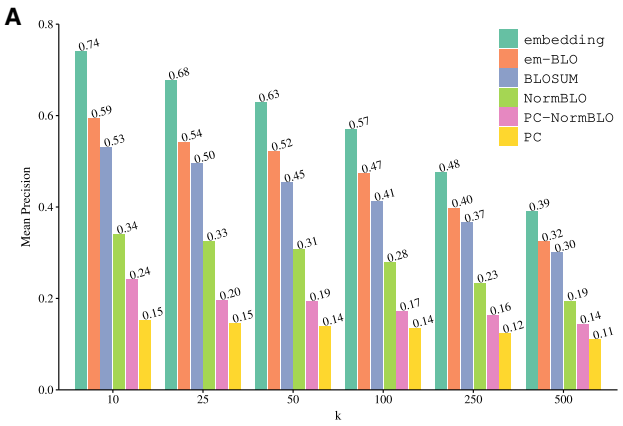
Amino acid embedding: Representation method

In addition to the embedding methods evaluated above in "amino acid embedding: learned versus hard-coded features," we compared the performance of MHCrank's embedding method against the performance garnered by leveraging the embedding methods of DeepLigand (Zeng and Gifford, 2019) and MHCSeqNet (Phloyphisut et al., 2019) within the MHCrank framework. These models are state-of-the-art MHC class I binding affinity prediction models. Both DeepLigand and MHCSeqNet were selected because their embedding methods are rooted in natural language processing, an approach which tends to provide robust embeddings.

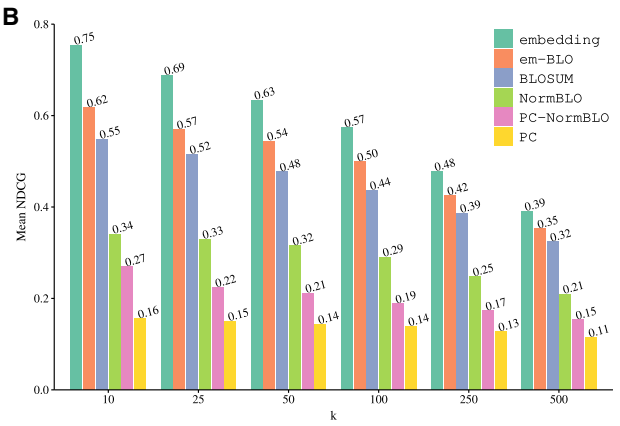
DeepLigand's embedding method is an "Embeddings from Language Model" (ELMo) (Zeng and Gifford, 2019). ELMo applies a deep bidirectional language model to the amino acids of a peptide, which are treated as words in a sentence. This enables ELMo to learn a context-dependent embedding for each amino acid. That is, the embedding returned for any given amino acid will depend not only on the amino acid itself but also on its position in the peptide as well as the surrounding residues. Because both DeepLigand and MHCrank make predictions regarding peptides presented by MHC class I molecules, both models share a corpus of peptides. As such, when leveraging the ELMo embedding method, we used a pre-trained implementation that output 64-dimension embedding vectors.

MHCSeqNet's embedding method, which is based on the skip-gram model with a window of 3 (Phloyphisut et al., 2019) (3-gram), also attempts to learn a context-dependent embedding for amino acids. Unlike ELMo, 3-gram does not consider the amino acid's position in the peptide. Rather, it only considers the amino acid itself and the identity of its surrounding residues. Specifically, 3-gram uses a window size of three. That is, the embedding for each amino acid is dependent on the previous two residues. When leveraging 3-gram within the MHCrank framework, we use a pre-trained embedding layer that outputs 100-dimension embedding vectors.

The results of this experiment are presented in Table S5. Using the embedding method within the MHCrank framework achieved significantly better performance than using either ELMo or 3-gram. A potential reason for this substantial difference in performance may be that ELMo and 3-gram do not fit well within the MHCrank framework. This is because the convolution operations that follow the embedding layer in MHCrank are aimed at learning contextual



Mean precision @ k for amino acid embedding methods tested for use in MHCrank.

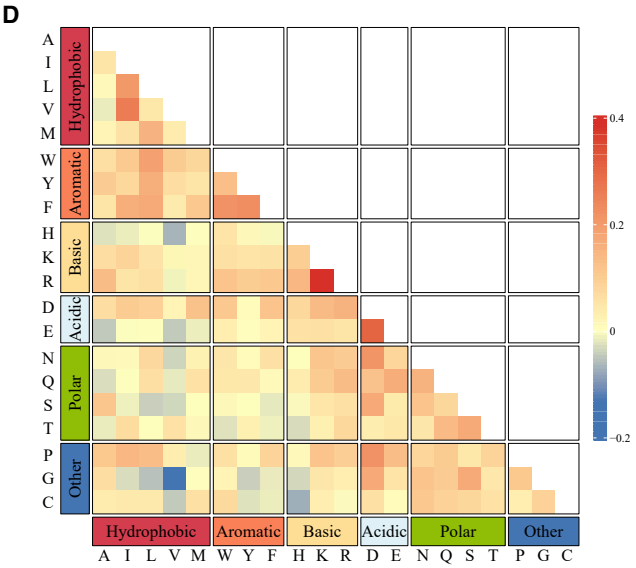


Mean NDCG @ k for amino acid embedding methods tested for use in MHCrank.

C

Embedding Method	Mean AUC ±95% CI
embedding	0.912±4.0e-4
em-BLO	0.912±3.8e-4
BLOSUM	0.912±3.8e-4
NormBLO	0.891±3.8e-4
PC-NormBLO	0.852±4.4e-4
PC	0.827±4.8e-4

Mean AUC for amino acid embedding methods tested for use in MHCrank.



Cosine similarity matrix of amino acid embeddings learned by MHCrank.

Figure 3. Performance of MHCrank with differing amino acid embedding methods

(A–C) Mean precision and NDCG@k and AUC, respectively for six identical architectures of MHCrank each trained while using a distinct amino acid embedding method. The embedding method is used by all the best-performing MHCrank ensembles, including Fw-top2. NormBLO refers to a normalized version of the BLOSUM matrix; PC is the embedding matrix we produce using physicochemical properties (see "amino acid representation").

(D) Cosine similarities of embeddings learned by an MHCrank model randomly selected from the best-performing ensemble (Fw-top2).

See also Tables S5 and S6.

dependencies, which affect the likelihood of a peptide being processed for presentation. Thus, embedding methods such as ELMo and 3-gram, which already provide context-dependent embeddings, will be in conflict with these convolution layers, and as a result MHCrank will not learn well.

Amino acid embedding: MHCrank embedding dimension parameter study

After determining that embedding is the optimal embedding method among those evaluated in Figure 3 and Table S5 to use within the MHCrank framework, we preformed a parameter study to identify the embedding's optimal dimensions. We evaluated the performance of MHCrank when the dimensions of the embedding method were set at 10, 20, 21, 50, and 100. Table S6 presents the results of these experiments.

The AUC for each dimension evaluated is reported in Table S6A. We found that embedding dimensions of 10 and 100 achieved the best AUC, but dimensions of 20 and 21 achieved very similar AUC (only 0.4% difference). Moreover, in terms of precision@k, the results reported in Figure S6B highlight

that using the embedding method with 20 or 21 dimensions achieved significantly better performances than when either of the other three dimensions were used. Note that the performances garnered from using either 20 or 21 dimensions were not significantly different. This is unsurprising given how similar the dimensions are. We found that using 10 dimensions yielded the next best performance. While similar in its dimensions, the reduction in performance from using the 10-dimension embedding method likely results from diminished expressiveness. The performance attained using embedding of dimension 50 is significantly reduced compared with dimensions 10, 20, and 21. Finally, across all values of k, the use of 100 dimensions achieved the worst performance. The successive reduction in performance from 20/21 dimensions to 50 and again from 50 dimensions to 100 is likely the result of overparameterization. That is, the number of parameters that need to be learned is greater than is allowed for by the size of the training data. The trends in performance for precision@k are identical to those observed for NDCG@k as well. Therefore, because the AUC performance obtained by the embedding method with 20 and 21 dimensions is very similar to the best-performing dimensions and these dimensions garnered the best performance both for precision@k and NDCG@k, the results of this parameter study indicate that using embedding with 20 or 21 dimensions is optimal given the current MHCrank framework and the training data used.

Case studies using SARS-CoV-2 data

In Figure S4, we present two case studies using SARS-CoV-2 data (Kared et al., 2021; Snyder et al., 2020). These case studies were conducted to evaluate MHCrank's viability in a real-world setting. Both of the studies report CD 8^+ T cell receptors (TCR) responses of various alleles to candidate peptides. Each discussed method employs a distinct process to determine candidates.

Kared et al.

This study (Kared et al., 2021) reports the TCR responses of 142 specific peptide-allele combinations from 30 active and convalescent SARS-CoV-2 patients. Here, peptide candidates are determined by obtaining a consensus of MHC class I prediction models while considering homology with experimentally validated antigens from other viruses in the SARS-CoV family. Practically, this ensures that the candidates selected will be more representative of the most highly immunogenic peptides. This is because each method learns a different set of features surrounding either a peptide's likelihood to be processed or its binding affinity in comparison with other models. Furthermore, the candidates identified through a consensus will have demonstrated many characteristics that increase presentation favorability. During their experiments, the authors denote hits as the number of TCRs identified that were specific for peptides within the patient cohort. Higher hits indicate that this combination is more immunogenic. That is, the combination will drive CD8^+ T cell responses to a greater extent.

Because antigen processing is aimed at producing peptides to be presented by MHC class I molecules, we hypothesize that MHCrank's predictions could be used as a predictor of a peptide's immunogenicity. As such, we selected both netMHCpan4.1-BA and netMHCpan4.1-EL as baselines and obtained predictions for each peptide-allele combination from the models. Because MHCrank is allele agnostic, we grouped the peptide-allele combinations to obtain the number of hits for each peptide. Note that we do not use these groupings for netMHCpan4.1-BA and netMHCpan4.1-EL, as the scores from both models are allele dependent.

Given that the goal of vaccine development is to identify which peptide(s) will elicit the best immune response, we wanted to identify which model's predicted scores were most closely correlated with the number of hits. High levels of correlation would enable a model's predicted scores to be used as a measure of relative immunogenicity. In Figures S4A–S4C, we plot the score (x axis) for each of 46 peptides against their respective $\log_2(\text{hits})$ and fit a line to the data. Figure S4G reports the slope of the regression lines as well as the Pearson correlation and respective p value for each model.

In Figure S4G, we observed that MHCrank achieved a 451% increase in the slope of the regression line over netMHCpan4.1-BA. Additionally, the correlation coefficient calculated for MHCrank is 966% higher than netMHCpan4.1-BA's correlation coefficient. Furthermore, the correlation between netMHCpan4.1-BA's predicted binding affinity and the number of hits is not statistically significant from an uncorrelated system. We also find that MHCrank also achieves an 18% increase in its slope of regression over netMHCpan4.1-EL's slope and a 76% increase in its correlation with hits over the netMHCpan4.1-EL model. Note that, like netMHCpan4.1-BA, the correlation reported for netMHCpan4.1-EL is not significant.

The difference in the distributions of points in Figures S4B and S4C provides a possible explanation. netMHCpan4.1-EL predicts that the likelihood of a peptide to be processed will be above 0.8 for the majority of peptide-allele combinations. Comparatively, MHCrank provides a much greater range in predicted scores that is more closely associated with a peptide's T cell response. Practically, this observation supports MHCrank's use over netMHCpan4.1-EL. While it is important to be able to identify which peptides will be presented, being able to predict which peptides will elicit the strongest immune response and CD8^+ T cell production would have greater utility in streamlining vaccine and drug development.

Snyder et al.

Our second SARS-CoV-2 case study utilizes data from Snyder et al. (2020). These data measured TCR responses to approximately 550 candidate peptides from over 1,500 Covid-19 patients. Unlike the data presented by Kared et al., the candidates identified here are exclusively those with the highest 2% of binding affinities, as predicted by netMHCpan4.1-BA. Overlapping peptides were combined into "antigen groups" by the authors. There were 260 antigen groups in total. The number of hits reported for each group indicates the number of TCRs that were specific for any of the peptides in said group. Similar to the Kared et al. data, a higher number of hits indicates an increase in a peptide's immunogenicity.

To evaluate each predictor's ability to estimate, we obtained predictions for each peptide using netMHCpan4.1-BA, netMHCpan4.1-EL, and MHCrank. For each antigen group, the peptide with the highest predicted score was used to represent the entire

region. This was done because there was no indication as to the proportion or specific number of hits corresponding to individual peptides. In Figures S4D–S4F, we plot the score (x axis) for each antigen group against their respective $\log_2$(hits) and fit a line to the data. Figure S4H reports the slope of the best-fit lines as well as the Pearson correlation and their respective p value for each model.

We notice a substantial increase in the correlation between netMHCpan4.1-BA's predicted binding affinity and $\log_2$(hits) when using the Snyder et al. data compared with when using the Kared et al. data (Figures S4H and S4G, respectively). Unlike when the Kared et al. data are used, both netMHCpan4.1-BA and netMHCpan4.1-EL exhibit significantly higher correlations with the hits from Snyder et al. than a random model. MHCrank is slightly outperformed by netMHCpan4.1-BA yet still outperforms netMHCpan4.1-EL. Additionally, its correlation remains consistent across both datasets. This highlights not only MHCrank's versatility as an estimator of peptide immunodominance but also a potential issue with the approach employed by Snyder et al.

First, grouping candidate peptides into antigen groups inhibits the ability to judge individual candidate rankings. That is, by summing all the hits belonging to overlapping peptides, the importance of the region to TCR response is evaluated rather than the immunogenicity of specific peptides. This is because it is unlikely that peptides within each antigen group would have the same number of hits, nor would they necessarily bind to the same TCRs. Thus, while netMHCpan4.1-BA and netMHCpan4.1-EL perform reasonably well in the test conditions offered by the Snyder et al. dataset, it is possible that since candidates are filtered such that they comprise the top 2% of netMHCpan4.1-BA's predicted binders in the SARS-CoV-2 proteome, if the peptides were not collapsed into antigen groups the distributions of $\log_2$(hits) to netMHCpan4.1-BA and netMHCpan4.1-EL's predicted scores would more closely resemble that which was observed in Figures S4A and S4B, respectively—in which case, the correlation would be substantially reduced as well.

This also raises an issue with the processes used to select and identify candidates. By only using netMHCpan4.1-BA to identify candidates, Snyder et al. not only introduce sampling bias but also control against any other factors affecting binding affinity that have not been learned by netMHCpan4.1-BA. Importantly, this includes the conditions that affect processing favorability. As a result, the candidates identified by Kared et al. are more likely to represent a higher proportion of the total number of antigens to which the TCRs are specific. Furthermore, this presents a plausible reason for netMHCpan4.1-BA's extremely poor performance when candidates are not limited by its perception of binding affinity.

Finally, the consistency in MHCrank's performance demonstrates that while netMHCpan4.1-BA may be slightly more adept at identifying important regions for stimulating TCR responses, MHCrank performs superiorly at estimating the immunogenicity of individual peptides. However, for vaccine and drug development, being able to identify the best peptides is arguably more important than the regions from which they originate.

DISCUSSION

We observed that all but two of the models included in the MHCrank ensembles processed peptides to a length of 9 or 10 amino acids, and Figures 3A–3C demonstrate that MHCrank models trained using embedding amino acid representation significantly outperform all other evaluated embedding methods. These findings indicate that peptide representations are improved through the enhanced biological relevance of our pre-processing method and the inclusion of learned embeddings. For pre-processing, we believe that our leveraging the knowledge that central amino acids of longer peptides often do not interact with the MHC class I molecule enabled processed peptides to retain only the most relevant information. The dearth of enrichment in the central amino acids both in the training peptides Figure 2A and in the top 100 ranked peptides from MHCrank (Figure 2B), paired with our improved performance over MHCflurry-AP (Table S2D), further strengthens the claim that central amino acids are not necessarily relevant features and that their removal may improve model performance.

With respect to the unique embedding learned for each amino acid, the enrichment we observed in Figure 2 also highlights their capability to identify features and information that may not have been imparted by the BLOSUM matrix. As observed in Figure 2B, all but one of the hydrophobic amino acids are enriched to comparable levels at the C-terminal position in Fw-top2's top predicted peptides. Not only does this coincide with biological observations but, as we observed in Figure 3D, it also suggests that MHCrank has learned to identify, and favor, certain physiochemical properties of amino acids, such as hydrophobicity, despite no a priori knowledge (Comber and Philip, 2014). Furthermore, the relative enrichment of MHCrank's ninth position versus its seventh and eighth positions (Figure 2B) suggests that the implementation of the cleavage site-specific kernel aids in the identification of commonalities among protease cleavage site motifs. This is apparent when considering the enrichment of the same positions from peptides ranked by MHCflurry-AP.

Thus, even for different numbers of top-*k* ranked peptides based on predictions, where the performance of MHCrank and netMHCpan were not significantly different, we believe MHCrank might still be considered superior because it achieves better performance using more unique peptides. MHCrank also achieved superior performance compared with MHCflurry-AP for all evaluated metrics. In addition, the amino acid learned embeddings and enrichment were highly correlated with biological observations. Altogether, the proposed MHCrank demonstrates strong performance compared with existing methods, and could have vast applicability in aiding drug and vaccine development.

Rationale for benchmarking against netMHCpan version 4.0 over version 4.1

It should be noted that while we utilize netMHCpan4.0-EL as one of our baselines, a newer version (4.1) (Reynisson et al., 2020) has been released. We considered both versions and elected to use version 4.0 as our baseline for the following specific reasons. First, while methodologically both versions are quite similar, a key distinction is the amount of training data used to train each model (Jurtz et al., 2017; Reynisson et al., 2020).

Specifically, netMHCpan4.1-EL was trained on over 13 million peptides. Comparatively, both netMHCpan4.0-EL and MHCrank were trained on datasets that are less than 10% of this size. Such large differences in the number of observations make it difficult to ascertain whether improvements arise from minor methodological differences or from the inclusion of additional training data. As such, it is considered unfair to compare the performance of methods if their models are trained over substantially different datasets. Therefore, we believe that netMHCpan4.0-EL allows for a more accurate and fair comparison.

There were also concerns regarding a substantial overlap in positive instances of netMHCpan4.1-EL's training data and MHCrank's testing data. When evaluating model performance, it is imperative that none of the testing data be in the training data, which can artificially elevate a model's performance. However, when comparing netMHCpan-4.1's reported training data (found here: https://tinyurl.com/netMHCpan41Data), we identified an overlap of over 100,000 peptides between netMHCpan-4.1's training data and our testing data. Furthermore, because approximately 60% of MHCrank's testing hits are present in netMHCpan-4.1's training data, any direct comparison using MHCrank's testing data may not be accurate. In fact, the performance reported for netMHCpan4.1-EL is likely inflated.

Furthermore, netMHCpan4.1-EL's implementation is not reproducible. That is, the source code is unavailable and the parameters of their model (e.g., number of layers, dimensionality of layers) are not discussed. This prevents us from reproducing their model and training it on our own data. netMHCpan4.0-EL was determined to be the better option to use as a baseline because we could, at a minimum, guarantee that there would be no overlap in the training and testing data sets. This is because the entirety of the data we used for testing was released after netMHCpan4.0-EL was published.

We do acknowledge that a comparison of MHCrank with this new version may still be desired. As such, we still report the percent improvement of MHCrank relative to netMHCpan4.1-EL in Table S4, where we used the command line, pre-trained, software version of netMHCpan4.1-EL to do the prediction. While the performance of MHCrank is not significantly better than that of netMHCpan4.1-EL for $k > 10$, the performance at $k = 10$ still supports the use of MHCrank. This is because during vaccine and drug development, only a handful of peptides will be selected and optimized. As such, it is more valuable to have high performance at lower values of k. Furthermore, the reduced performance at higher levels of k could be influenced by the discrepancy in amount of data each model is trained on. That is, while MHCrank better identifies the peptides that are most likely to be processed for presentation, the increased data made available to netMHCpan4.1-EL may have allowed it to learn feature differences between the peptides that are less likely to be processed and the less trivial decoys. As the more difficult decoys are exhausted, the performance of MHCrank relative to netMHCpan4.1-EL rises again. Additionally, netMHCpan4.1-EL's exposure to the majority of positive testing instances during training may also account for the dip in MHCrank's performance relative to netMHCpan4.1-EL from $k = 25$ to $k = 500$ that was observed in Table S4. Thus, even when MHCrank and netMHCpan4.1-EL are compared, the foregoing discussion supports the notion that MHCrank is better at ranking the peptides most likely to be processed at the very top.

Limitations of the study

Despite MHCrank's strong performance, there are multiple improvements that might further strengthen its relevance and applicability for drug and vaccine development. First, the implementation of a combinatorial approach that incorporates binding affinity as well—similar to O'Donnell et al.'s presentation score predictor (O'Donnell et al., 2020)—might improve upon the current presentation predictions achieved for the MHC class I molecule.

Second, given that the purpose of predicting MHC presentation is to aid in the development of drugs and vaccines that stimulate the adaptive immune response through T cell activation, future endeavors may benefit from predicting both the magnitude and type of response a specific antigen will elicit. For MHC class I presented antigens in particular, this might be accomplished by training models to identify complementary sequences between the presented peptide's central amino acids and the TCR's active site to which it binds.

Finally, MHCrank and other MHC binding prediction models do not effectively utilize protein structure for predictions. While BLOSUM62 (Henikoff and Henikoff, 1992) and other popular embedding schemes aim to encapsulate sequence homology and (dis)similarities among amino acids, this is a suboptimal approach. The sequence of a peptide is not, in and of itself, deterministic of an antigen's ability to be processed, presented, and recognized. Rather, it is a product of physicochemical properties that result from individual interactions of amino acids (Wieczorek et al., 2017). The utilization of structural data requires a structure to first be resolved in a lab setting, rendering the approach infeasible. However, as structural prediction algorithms improve and become increasingly biologically relevant, abandoning sequences for structures will likely improve model performance. Specifically, the use of geometric DL models seems poised to yield the highest probability of success (Gainza et al., 2020). We will investigate along these lines in our future research.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Training data
  - Testing data processing
  - Baseline methods
  - MHCrank methods
  - Peptide pre-processing

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.crmeth.2022.100293.

ACKNOWLEDGMENTS

This project was made possible, in part, by support from the National Institute of General Medical Sciences (2R01GM118470-05) and an AWS Machine Learning Research award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

AUTHOR CONTRIBUTIONS

Conceptualization, X.N.; methodology, X.N. and P.J.L.; software, P.J.L.; validation, X.N. and P.J.L.; formal analysis, X.N. and P.J.L.; investigation, X.N. and P.J.L.; resources, X.N.; data curation, P.J.L.; writing – original draft, P.J.L.; writing – review & editing, X.N. and P.J.L.; visualization, P.J.L. and X.N.; supervision, X.N.; project administration, X.N.; funding acquisition, X.N.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: December 9, 2021
Revised: March 17, 2022
Accepted: August 19, 2022
Published: September 19, 2022

REFERENCES

Abele, R., and Tampé, R. (2004). The ABCs of immunology: structure and function of TAP, the transporter associated with antigen processing. Physiology *19*, 216–224. https://doi.org/10.1152/physiol.00002.2004.

Abelin, J.G., Harjanto, D., Malloy, M., Suri, P., Colson, T., Goulding, S.P., Creech, A.L., Serrano, L.R., Nasir, G., Nasrullah, Y., et al. (2019). Defining HLA-II ligand processing and binding rules with mass spectrometry enhances cancer epitope prediction. Immunity *51*, 766–779.e17. https://doi.org/10.1016/j.immuni.2019.08.012.

Boehm, K.M., Bhinder, B., Raja, V.J., Dephoure, N., and Elemento, O. (2019). Predicting peptide presentation by major histocompatibility complex class I: an improved machine learning approach to the immunopeptidome. BMC Bioinf. *20*, 7. https://doi.org/10.1186/s12859-018-2561-z.

Center, O.S. (1987). Ohio supercomputer center. http://osc.edu/ark:/19495/f5s1ph73.

Chen, Z., Min, M.R., and Ning, X. (2021). Ranking-based convolutional neural network models for peptide-MHC class I binding prediction. Front. Mol. Biosci. *8*, 634836. https://doi.org/10.3389/fmolb.2021.634836.

Comber, J.D., and Philip, R. (2014). MHC class I antigen presentation and implications for developing a new generation of therapeutic vaccines. Ther. Adv. Vaccines *2*, 77–89. https://doi.org/10.1177/2051013614525375.

Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Boscaini, D., Bronstein, M.M., and Correia, B.E. (2020). Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nat. Methods *17*, 184–192. https://doi.org/10.1038/s41592-019-0666-6.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning (The MIT Press).

Guo, H.C., Jardetzky, T.S., Garrett, T.P., Lane, W.S., Strominger, J.L., and Wiley, D.C. (1992). Different length peptides bind to HLA-Aw68 similarly at their ends but bulge out in the middle. Nature *360*, 364–366. https://doi.org/10.1038/360364a0.

Henikoff, S., and Henikoff, J.G. (1992). Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA *89*, 10915–10919. https://doi.org/10.1073/pnas.89.22.10915.

Jurtz, V., Paul, S., Andreatta, M., Marcatili, P., Peters, B., and Nielsen, M. (2017). NetMHCpan-4.0: improved peptide–MHC Class I interaction predictions integrating eluted ligand and peptide binding affinity data. J. Immunol. *199*, 3360–3368. https://doi.org/10.4049/jimmunol.1700893.

Kared, H., Redd, A.D., Bloch, E.M., Bonny, T.S., Sumatoh, H., Kairi, F., Carbajo, D., Abel, B., Newell, E.W., Bettinotti, M.P., and Benner, S.E. (2021). SARS-CoV-2–specific $CD8^+$ T cell responses in convalescent COVID-19 individuals. J. Clin. Investig. *131*. https://doi.org/10.1172/JCI145476.

Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T., and Kanehisa, M. (2008). AAindex: amino acid index database, progress report 2008. Nucleic Acids Res. *36*, D202–D205. https://doi.org/10.1093/nar/gkm998.

Kloetzel, P.M. (2001). Antigen processing by the proteasome. Nat. Rev. Mol. Cell Biol. *2*, 179–187. https://doi.org/10.1038/35056572.

Lundegaard, C., Lund, O., Buus, S., and Nielsen, M. (2010). Major histocompatibility complex class i binding predictions as a tool in epitope discovery. Immunology *130*, 309–318. https://doi.org/10.1111/j.1365-2567.2010.03300.x.

O'Donnell, T.J., Rubinsteyn, A., and Laserson, U. (2020). MHCflurry 2.0: improved pan-allele prediction of MHC Class I-presented peptides by incorporating antigen processing. Cell Syst. *11*, 42–48.e7. https://doi.org/10.1016/j.cels.2020.06.010.

Phloyphisut, P., Pornputtapong, N., Sriswasdi, S., and Chuangsuwanich, E. (2019). MHCSeqNet: a deep neural network model for universal MHC binding prediction. BMC Bioinf. *20*, 270. https://doi.org/10.1186/s12859-019-2892-4.

Reynisson, B., Alvarez, B., Paul, S., Peters, B., and Nielsen, M. (2020). 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. Nucleic Acids Res. *48*, W449–W454. https://doi.org/10.1093/nar/gkaa379.

Rock, K.L., and Goldberg, A.L. (1999). Degradation of cell proteins and the generation of mhc class I-presented peptides. Annu. Rev. Immunol. *17*, 739–779. https://doi.org/10.1146/annurev.immunol.17.1.739.

Rock, K.L., York, I.A., and Goldberg, A.L. (2004). Post-proteasomal antigen processing for major histocompatibility complex class I presentation. Nat. Immunol. *5*, 670–677. https://doi.org/10.1038/ni1089.

Sarkizova, S., Klaeger, S., Le, P.M., Li, L.W., Oliveira, G., Keshishian, H., Hartigan, C.R., Zhang, W., Braun, D.A., Ligon, K.L., et al. (2020). A large peptidome dataset improves HLA class I epitope prediction across most of the human population. Nat. Biotechnol. *38*, 199–209. https://doi.org/10.1038/s41587-019-0322-9.

Shraibman, B., Barnea, E., Kadosh, D.M., Haimovich, Y., Slobodin, G., Rosner, I., López-Larrea, C., Hilf, N., Kuttruff, S., Song, C., et al. (2019). Identification of

tumor antigens among the HLA peptidomes of glioblastoma tumors and plasma. Mol. Cell. Proteomics *18*, 1255–1268. https://doi.org/10.1074/mcp.RA119.001524.

Snyder, T.M., Gittelman, R.M., Klinger, M., May, D.H., Osborne, E.J., Taniguchi, R., Zahid, H.J., Kaplan, I.M., Dines, J.N., Noakes, M.T., et al. (2020). Magnitude and dynamics of the T-cell response to SARS-CoV-2 infection at both individual and population levels. Preprint at medRxiv. https://doi.org/10.1101/2020.07.31.20165647.

Wagih, O. (2017). ggseqlogo: a versatile R package for drawing sequence logos. Bioinformatics *33*, 3645–3647. https://doi.org/10.1093/bioinformatics/btx469.

Wieczorek, M., Abualrous, E.T., Sticht, J., Álvaro-Benito, M., Stolzenberg, S., Noé, F., and Freund, C. (2017). Major histocompatibility complex (MHC) class i and MHC class II proteins: conformational plasticity in antigen presentation. Front. Immunol. *8*, 292. https://doi.org/10.3389/fimmu.2017.00292.

Yewdell, J.W., Reits, E., and Neefjes, J. (2003). Making sense of mass destruction: quantitating MHC class I antigen presentation. Nat. Rev. Immunol. *3*, 952–961. https://doi.org/10.1038/nri1250.

Zeng, H., and Gifford, D.K. (2019). DeepLigand: accurate prediction of MHC class I ligands using peptide embedding. Bioinformatics *35*, i278–i283. https://doi.org/10.1093/bioinformatics/btz330.

CellPress
OPEN ACCESS

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Training data for MHCrank ensembles	(O'Donnell et al., 2020)	https://doi.org/10.17632/zx3kjzc3yx.3 (see Data S5)
Multialleleic benchmark dataset	(O'Donnell et al., 2020)	https://doi.org/10.17632/zx3kjzc3yx.3 (see Data S1)
Kared et al. SARS-CoV-2 dataset	(Kared et al., 2021) supplemental data	https://doi.org/10.1172/JCI145476
Snyder et al. SARS-CoV-2 dataset	(Snyder et al., 2020)	https://doi.org/10.21417/ADPT2020COVID
Software and algorithms		
Python	Python Software Foundation	v3.6.6
Tensorflow	Google	v2.2.1
MHCrank	This paper	https://doi.org/10.5281/zenodo.6999030
MHCflurry-2.0	(O'Donnell et al., 2020)	https://github.com/openvax/mhcflurry
netMHCpan-4.0	(Jurtz et al., 2017)	https://services.healthtech.dtu.dk/service.php?NetMHCpan-4.0
netMHCpan-4.1	(Reynisson et al., 2020)	https://services.healthtech.dtu.dk/service.php?NetMHCpan-4.1
DeepLigand	(Zeng and Gifford, 2019)	https://github.com/gifford-lab/DeepLigand
MHCSeqNet	(Phloyphisut et al., 2019)	https://github.com/cmb-chula/MHCSeqNet
Other		
CPU	Ohio Supercomputer Center	Intel Xeon 8268s Cascade Lakes
GPU	Ohio Supercomputer Center	NVIDIA Volta V100

RESOURCE AVAILABILITY

Lead contact

Further information and requests should be directed to and will be fulfilled by the lead contact, Xia Ning (ning104@osu.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- This paper analyzes existing, publicly available data. These accession numbers for the datasets are listed in the key resources table.
- All original code as been deposited at https://github.com/ninglab/mhcrank and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in the paper is available from the lead contact upon request.

METHOD DETAILS

Training data

Our training data set was identical to that which was used by O'Donnell et al. to train their MHCflurry-AP predictor (O'Donnell et al., 2020). This data set was compiled from two studies (Abelin et al., 2019; Sarkizova et al., 2020) and comprised the aggregate data from 100 mass spectroscopic (MS) experiments, including measurements on 8,537,960 distinct peptides and 92 different HLA alleles. Based on the nature of the MS experiments, bound peptides (hits) must have first underwent processing for MHC Class I presentation. Hits with a sequence length between eight and fifteen amino acids were selected as this range is optimal for presented peptides. O'Donnell et al. randomly generated 99 negative decoys per hit. Each decoy was the same length and was extracted from the same protein as the hit to which it corresponds. O'Donnell et al. used their binding affinity predictor to select decoys most similar to hits in terms of their predicted binding affinities. The hits and decoys selected were within the top 2% of predicted binding affinities for hits and decoys, respectively. The exclusion of weak binding hits and decoys from the data set was aimed at facilitating the model's ability to learn features that strongly influence a peptide's likelihood to be processed for presentation as opposed to learn features

associated with binding affinity. This yielded 399,392 peptides in the training data, 44% of which were hits (O'Donnell et al., 2020). The data was split into 4 training subsets (folds) by randomly withholding 10 MS experiments from each. As a result, the number of samples in each fold was 365,746; 352,144; 361,864 and 358,374, respectively. Additionally, each fold had 10% of its samples randomly withheld for validation.

Testing data processing

We used the same testing set to evaluate our MHCrank model as was used by O'Donnell et al. to evaluate their model (O'Donnell et al., 2020). The data in the testing data set was compiled from 2 studies published in 2019; these studies comprised 20 experiments and identified 27,007 binding peptides (Sarkizova et al., 2020; Shraibman et al., 2019). According to O'Donnell et al., these specific experiments were withheld for testing as they were not yet published when their baseline methods, such as netMHCpan, were created (O'Donnell et al., 2020). Therefore, none of the models being evaluated would have been exposed to the testing data for their training. 99 decoys were introduced for each hit in the testing set, bringing the total number of testing peptides up to 2,700,700. The testing set was also used by O'Donnell et al. to benchmark their binding affinity predictors. This required each peptide to be paired with multiple, distinct HLA alleles. However, for HLA-independent models, such as MHCrank and our baselines, which do not use HLA allele information, these distinct combinations are instead interpreted as duplicated samples that achieve identical scores. As a result, duplicated samples dominate the results and can either negatively or positively bias ranking performance. To prevent this, we removed any duplications of a peptides from the test set, leaving a remaining 2,409,183 peptides. Approximately, 0.73% of the remaining peptides were hits.

Baseline methods

MHCrank was compared to two baseline methods: MHCflurry-AP and netMHCpan4.0-EL. Both the baselines, similar to MHCrank, are AP prediction models. MHCflurry-AP was selected for its high performance. Note that because MHCrank architectures were leveraged from MHCflurry-AP, by comparing MHCrank to MHCflurry-AP, any improvements in performance exhibited by MHCrank may be associated with the introduced architectural alternations. The other baseline, netMHCpan4.0-EL, was selected as it was identified by O'Donnell et al. to be the best performing AP predictor available (O'Donnell et al., 2020). netMHCpan4.0-EL utilizes an ensemble of neural networks with 2 hidden layers that output predictions of both binding affinity and eluted ligands (Jurtz et al., 2017). Given the focus of our study was to improve predictions of which peptides are more favorable to be processed, we used only the eluted ligand output from netMHCpan4.0-EL. The predictions we used to evaluate the performance of MHCflurry-AP and netMHCpan4.0-EL were those reported by O'Donnell et al. (O'Donnell et al., 2020).

MHCrank methods

The MHCrank architecture is presented in Figure 1. The proposed MHCrank takes three types of information as input: 1) uniform-length sequence N-flank + peptide + C-flank, 2) C-flank cleavage site sequence, and 3) the peptide's original length. The N- and C-flanks are defined as the first five amino acids adjacent to a peptide on its amine (N-) and carboxyl (C-) terminal ends, respectively.

The C-flank cleavage site sequence is comprised of the terminal r amino acids of the peptide and the initial r amino acids of the C-flank. In the following subsections, we present how MHCrank learns from a peptide and its N- and C-flanks to predict antigen processing.

Peptide pre-processing

As in other methods, all input N-flank + peptide + C-flank sequences are first processed to obtain uniformity in length before being passed to MHCrank. Peptides of various lengths (8–15 amino acids as in our data set) can bind to MHC Class I molecules because only their termini interact with and anchor to the molecules (Rock et al., 2004). The central amino acids of a bound peptide create an arch-like formation as the peptide bows increasingly away from the pocket with increased length (Guo et al., 1992). As the central amino acids do not contribute directly to binding, their inclusion in MHCrank may be uninformative and ultimately detract from the model's predictive ability. Thus, unlike other methods, the sequence processing and representation in MHCrank was designed to favor the amino acids near an antigen's termini – it unifies the sequences of various lengths to length 10 (MHC Class I molecules favor peptides with 8–10 amino acids). Figure S1 presents the antigen representation process.

Specifically, if the peptide sequence is shorter than 10 amino acids, additional pseudo amino acids, represented by an ambiguous 'X', are added at the center positions of the peptide S1a. When the peptide's original length is odd, one amino acid is padded offset towards theN-terminal side. This was motivated by past studies demonstrating that the C-terminal amino acids of a candidate antigen are more influential than the N-terminal amino acids with respect to whether or not a peptide will be processed for presentation (Kloetzel, 2001; Rock et al., 2004). If the peptide sequence is longer than 10 amino acids (Figure S1B), a number of amino acids will be trimmed from the center of the peptide sequence, with one amino acid offset from the center towards the N-terminal if necessary.

Amino acid representation

Once the peptide sequences are processed into uniform length, their remaining amino acids will be embedded as vectors to capture the sequence contents in a computer-readable representation. We evaluated MHCrank has using six distinct amino acid representation methods, denoted as BLOSUM, embedding, em-BLO, PC, NormBLO, and PC-NormBLO, respectively. We discuss the methods below:

BLOSUM In the BLOSUM method, each amino acid is represented by its corresponding 21-dimension vector extracted from the BLOSUM62 substitution matrix (Henikoff and Henikoff, 1992); this is the convention for amino acid representation. BLOSUM is abstracted from evolutionary similarities and difference of amino acids. This does not guaranteed the embeddings to be relevant for all tasks. Moreover, the presence of unrelated features could reduce model performance.

embedding, em-BLO Thus, in the embedding embedding method, we allowed MHCrank to learn the amino acid representations as a part of its training process. Each amino acid is first represented by an initial, random 21-dimension embedding vector; the embeddings will be learned (Chen et al., 2021) in MHCrank so as to maximize their presentation power and facilitate optimal performance. The intuition being that it may enable the identification of amino acid features that are more highly correlated with determining a peptide's favorability for MHC Class I processing. We selected 21 to be the dimensionality of each embedding vector as this is same dimensions as the BLOSUM. This was done so that any improvement garnered by learning the the amino acid embeddings is due to the data-driven, stronger expressiveness of the learned embeddings rather than excessive dimensionality. We concatenate these two methods in the em-BLO method to determine if the possess unique information that through their combination increases model performance. This yields a 42-dimension vector. Note that the padded pseudo amino acid, X, is represented as a zero vector by embedding.

PC, NormBLO, PC-NormBLO We also constructed a novel embedding method using physiochemical properties obtained from data in the AAIndex (Kawashima et al., 2008). We refer to this as the PC embedding method. Specifically, using this method, each amino acid is represented by an 8-dimension vector that corresponds to hydrophobicity, polarizability, isoelectric point, volume, molecular weight, sterics, helix probability, sheet probability. Given that physiochemical properties are more determinate of peptide binding, we thought that the information conveyed might be more relevant than the BLOSUM method. However, because the length of vectors produced by PC is much shorter than from BLOSUM, it is possible that it lacks the expressivity of BLOSUM, which could negatively impact performance.

To, account for this, we first normalized the values of the BLOSUM method to be within the same range as PC (NormBLO). The PC and NormBLO methods were then concatenated to produce the PC-NormBLO embedding method. BLOSUM was first normalized as its values were much larger than those of PC, which could result in the BLOSUM portion receiving more weight than PC when combined. We present results comparing various portions of the embedding methods in the following Results sections: 'Amino acid embedding: Learned vs. hard-coded features', 'Amino acid embedding: Representation method', and 'Amino acid embedding: MHCrank embedding dimension parameter study'.

MHCrank learning

Given the amino acid representations in a processed peptide, each peptide's embedding matrix is further padded, following MHCflurry-AP (O'Donnell et al., 2020), and forwarded to a 1-D convolutional layer with n_{k_1} kernels of size k, which aggregates different local information (k-mers) in the peptide. The output feature mapping is then forwarded to a number of components as described below that capture various signals from the training peptides and learn how each peptide's flanking regions affect the probability that a peptide will undergo antigen processing.

Convolution over N-flank and C-flank

As was done in MHCflurry, the portion pertaining to the N-flank sequence is extracted from the feature mapping output of the first 1-D convolution layer. Mean pooling is conducted over the N-flank's specific feature mapping to achieve the per-channel average for each amino acid in the N-flank. The results are then forwarded to a dense layer, which outputs a single value representing the flank's favorability as a cleavage site. Identical operations are applied to the C-flank sequence.

Convolution over peptide

Following MHCflurry-AP (O'Donnell et al., 2020), convolutions are also applied to the peptide to learn the relationship between the cleavage favorability of its N-terminus/C-terminus and the cleavage favorability of its central amino acids. The intuition is that peptides with a higher cleavage favorability at their terminal position relative to their central amino acids are more likely to be processed for presentation than peptides in which this is not the case. To learn this relationship near the N-terminus, the portion pertaining to the peptide sequence is extracted from the feature mapping output of the first 1-D convolution layer and is subsequently forwarded to two stacked 1-D convolutional layers. The first layer has n_{k_2} channels; the second layer has 1 channel. Both of the stacked convolutional layers employ kernels of size 1. The output of the second layer contains a score for each amino acid in the peptide that represents the residue's likelihood of being an N-terminal cleavage site. The scores corresponding to the N-terminal and C-terminal amino acids within the peptide (A in Figure 1) are forwarded to the downstream learning. A max pool is applied over

the non-N-terminal amino acids to identify the overall highest favorability for N-terminal cleavage to occur within the central amino acids. A max pool is also applied over the non-C-terminal amino acids to identify the overall highest favorability for C-terminal cleavage to occur within the central amino acids.

Convolution over peptide

A novel component of MHCrank is the global-kernel based convolution over the C-terminal cleavage site. The C-terminal cleavage site is comprised of the terminal r amino acids of the peptide and the initial r amino acids of the C-flank, where r is the cleavage radius. The global kernel, referred to as a cleavage site-specific kernel and denoted as CSSK, is applied on amino acid representations of the cleavage site sequence to capture global signals useful for cleavage and processing. This was motivated by the fact that the C-terminal end of the peptide is more influential than the N-terminal with respect to cleavage and processing (Kloetzel, 2001; Rock et al., 2004). Because the C-terminus is the primary anchor point between the antigen and MHC molecule during binding, explicitly learning from the C-terminal cleavage site could enable additional, useful signals to predict a peptide's favorability to be processed. The global kernel is used with the intuition that there are motifs located in this region, which are recognized by the peptidases and proteases that process peptides for presentation, that would be more easily learned and recognized by a global kernel.

Incorporating original peptide length

Compared to other methods, MHCrank has a novel, single node whose input is the peptide's original length before padding or trimming. The downstream convolution's use of the peptide's original length in MHCrank is motivated by the fact that peptide length is a significant contributing factor to both processing and presentation (Rock et al., 2004; Comber and Philip, 2014).

Combining all information

Two fully-connected layers are applied in succession to the concatenated output of all the above components. Each layers possess n_{k_2} nodes. The output is then forwarded to an output layer that predicts the probability of a peptide undergoing antigen processing. Peptides more likely to undergo antigen processing receive higher probabilities.

Ensemble methods and model selection

Ensembles (Goodfellow et al., 2016) have been an effective strategy in making more accurate predictions compared to that of a single model by reducing prediction variance. We developed the following six ensemble strategies to leverage and integrate multiple models, where the model performance on each fold was accessed using AUC score on each fold's validation data, and overall model performance was accessed using the average AUC score across all four folds.

- Fw-top1 (fold-wise top-1): for each fold, we identified its best model and then combined these models. That is, the final ensemble consists of 4 total models – 1 from each of 4 folds. Please note that 4 models may correspond to different hyperparameters.
- Fw-top2 (fold-wise top-2): for each fold, we identified its top-2 best models and then combined these models. That is, the final ensemble consists of 8 total models – 2 from each of 4 folds. Please note that 8 models may correspond to different hyperparameters.
- Ba-top1 (best average top-1): we first identified the best performing hyperparameter set that had the best average AUC over the 4 folds.

We then trained a model on each fold using the hyperparameter set. The final ensemble consists of 4 total models – 1 from each of 4 folds. Please note that these 4 models correspond to the same set of hyperparameters.

- Ba-top2 (best average top-2): we first identified the top-2 best performing hyperparameter sets that had the best average AUC over the 4 folds.

We then trained a model on each fold using the hyperparameter set. The final ensemble consists of 8 total models – 2 from each of 4 folds.

Please note that 4 models (1 from each of 4 folds) correspond to a single set of hyperparameters; the remaining 4 models (1 from each of 4 folds) correspond to another single set of hyperparameters.

Furthermore, the set of hyperparameters belonging to the first 4 models is distinct from the set of hyperparameters belonging to the second 4 models.

- C-top1 (combo top-1): we combined the Fw-top1 and Ba-top1 models. That is, the final ensemble consists of 8 total models, 4 from Fw-top1 and 4 from Ba-top1.
- C-top2 (combo top-2): we combined the Fw-top2 and Ba-top2 models. That is, the final ensemble consists of 16 total models, 8 from Fw-top2 and 8 from Ba-top2.

The predicted scores from the ensemble models are calculated as the mean of the predicted scores from each of its component models. Table S7A outlines the specific hyperparameter sets of the models selected for each of the ensembles. The hyperparameters listed in Table S7A are only those that underwent tuning. The full set of hyperparameter options used to construct our MHCrank models can be found in Table S7B. The training and validation AUC of the models selected for inclusion in each ensemble is detailed in Table S1.

Model training

Three distinct model variations, each corresponding to one of the embedding, BLOSUM, em-BLO amino acid representation methods, combined with each hyperparameter set, were trained on each of the four training data folds. More information on the amino acid representations can be found in the 'Amino acid representation' section. See 'Training data' section for details regarding how the data was split into folds for training and validation. All hyperparameters utilized are listed in Table S7B. Optimization was achieved using an Adam optimizer and a binary cross-entropy loss function. Models were trained for 500 epochs with an early stopping patience of 30 epochs.

Computing resources

Data processing and model training, validation, and testing were all executed on Pitzer clusters of the Ohio Supercomputer Center (1987). We implemented models using Python-3.6.6 and TensorFlow-2.2.1. We trained models with 1 Intel Xeon 8268s Cascade Lakes CPU node and 1 NVIDIA Volta V100 GPU totaling 32 GB of memory.

Hyperparameters

A total of 729 models were trained across each of the 4 folds and evaluated. Each model was produced using a unique combination of hyperparameters. The specific hyperparameter options evaluated is presented below in Table S7B. Note that while the table presents only 243 unique combinations, these would be applied to each of the 3 amino acid representation methods: embedding, BLOSUM, and em-BLO.

QUANTIFICATION AND STATISTICAL ANALYSIS

Evaluation metrics

Model were evaluated for ensemble selection by the AUC they achieved on validation folds. The fold and validation AUC for any model included in a MHCrank ensemble is presented in Table S1. MHCrank ensembles were compared to the MHCflurry-AP and netMHCpan4.0-EL baselines via AUC, precision@ k, and NDCG@ k for $k = \{10, 25, 50, 100, 250, 500\}$.

- AUC: It is the area underneath a receiver operating characteristic (ROC) curve designed for binary classification problems.

ROC a probability curve depicting true positive rates vs false positive rates over various prediction thresholds.
Thus, AUC measures how well a model is capable of distinguishing between two classes (e.g., hits and decoys).
Higher AUC values indicate better distinguishing capacity.

- Precision@ k: It measures the proportion of top-k ranked peptides that are also hits. It is calculated as follows,

$$\text{Precision@}k = \frac{R_k \cap H}{R_k},$$

where R_k is the set of top-k ranked peptides, and H is the set of hits.
Higher precision@ k scores indicate higher probabilities of correctly detecting hits within the top-k peptides.
Note that precision@ k is equivalent to the 'PPV' metric reported in O'Donnell et al. (O'Donnell et al., 2020).

- NDCG@ k: It is the normalized discounted cumulative gain (DCG) for top-k ranking.

DCG@ k is calculated as follows:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)},$$

where rel_i is the relevance of an peptide at position i indicating whether the recommended peptide is a hit (1) or a decoy (0); the numerator of the DCG@ k equation awards relevant peptides and punishes decoys; the denominator gives more weight to higher ranked recommendations.
NDCG@ k is the normalized DCG@ k.
Higher NDCG@ k scores indicates better performance.

Statistical analysis

We obtained 1,000 sets of 100,000 peptides via bootstrap resampling of our testing data. Processing likelihood scores were obtained for each of these peptides from both our ensembles and the baseline methods. For each sampling iteration, we calculate the AUC, precision@ k, and NDCG@ k performance metrics. We perform pairwise t-tests to compare the performance of each of the MHCrank ensembles against each baseline. Hypothesis correction was implemented as discussed below:

Multiple hypothesis correction

We perform multiple-hypothesis correction to determine the significance levels reported in both Tables 1, 2, 3, and S2. Specifically, we employ the Bonferroni method Given that we are making 6 pairwise comparisons: each of the 6 MHCrank ensembles compared against either netMHCpan4.0-EL or MHCflurry-AP (Tables 1, 2, and 3 and S2, respectively). the corrected significance thresholds are given by the following: $p \leq 0.1 : \ = \frac{0.1}{6}$; $p \leq 0.05 : \ = \frac{0.05}{6}$; $p \leq 0.1 : \ = \frac{0.01}{6}$; and $p \leq 0.1 : \ = \frac{0.001}{6}$.

Based on our analysis, there were very limited changes to the initially interpreted significance when using the Bonferroni correction. MHCrank retains significance for all the reported improvements over MHCflurry-AP and netMHCpan4.0-EL for AUC and over MHCflurry-AP for precision@ k and NDCG@ k for all values of k. Additionally, the use of a Bonferroni correction removes the significance from MHCrank's Fw-top2 ensemble at $k = 25$ for precision@k; MHCrank's C-top1 ensemble at $k = 50$ for NDCG@k; MHCrank's C-top2 ensemble at $k = 50$ for both precision@k and NDCG@k. However, because practically, interest lies in determining whether a given model outperforms another through direct comparison, in our case, multiple hypothesis correction may not be necessary for the evaluation of model performance. Furthermore, the SARS-CoV-2 case studies presented in the Results section 'Case studies using SARS-CoV-2 data' further demonstrates MHCrank's real-world utility over netMHCpan4.0-EL.

Site-specific amino acid enrichment

A set of 50,000 randomly selected hits were obtained from the training peptides. For each model, the set of the top 100-recommended peptides were selected. The peptides in each set were processed to a length of 9 amino acid residues via the method outlined in Figure S1 and the 'Peptide pre-processing' section. We produced a site-specific amino acid enrichment visualization from each set of peptides using the R package: ggseqlogo (Wagih, 2017).

Cosine similarity

Cosine similarity uses inner product space to measure the similarity between two vectors. The cosine similarity was calculated for the embedding vectors of each amino acid-pair and is given by the following:

$$\text{Cosine Similarity}(a,b) \ = \ 1 \ - \ \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \ \|\vec{b}\|},$$

where $\vec{a} \cdot \vec{b}$ is the dot product of two vectors. Higher values indicate higher similarities.