Prediction of peptide binding to MHC Class I proteins in the age of deep learning

Rohit Bhattacharya[1,2], Collin Tokheim[2,3], Ashok Sivakumar[1,2], Violeta Beleva Guthrie[2,3], Valsamo Anagnostou[4,5], Victor E. Velculescu[2,4,5], Rachel Karchin[2,3,4,*]

1 Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

2 Institute for Computational Medicine, Johns Hopkins University, Baltimore, MD, USA

3 Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA

4 The Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University School of Medicine, Baltimore, MD, USA

5 The Bloomberg-Kimmel Institute for Cancer Immunotherapy, Johns Hopkins University School of Medicine, Baltimore, MD, USA

∗ **E-mail: karchin@jhu.edu**

Abstract

Prediction of antigens likely to be recognized by the immune system is a fundamental challenge for development of immune therapy approaches. We explore the utility of deep learning for *in silico* prediction of peptide binding affinity to major histocompatibiliy complex Type I molecules (pMHC-I binding). This process is a critical step in the immune system's response to cancer cells, which may present highly specific neoantigen peptides bound to MHC proteins at the cell surface. With the advent of high-throughput sequencing and the recognition that somatic mutations in the exome can produce neoantigens, fast *in silico* prediction of these affinities has become increasingly relevant to precision cancer immunotherapy.

We have developed five machine learning methods and use a benchmark from the Immune Epitope Database of experimental pMHC-I binding affinities to compare them to existing machine learning approaches. All methods were used to score, rank, and classify pMHC-I pairs. The best six methods, which include three of our own, were identified and found to make highly correlated predictions, even for individual pMHC-I pairs. The most effective deep learning methods were a gated recurrent unit and a long short-term memory neural network, enhanced by transfer learning. These methods can handle peptides of any length without the need for artificial lengthening or shortening and were substantially faster than the most widely-used standard neural networks.

Major findings. The best *in silico* predictors of peptide major histocompatibility complex binding must be identified for application in precision cancer immunotherapy. We design and test a variety of machine learning methods for this purpose. We identify six best-in-class methods, three of our own design. Surprisingly, the best deep and standard machine learning methods make highly correlated predictions. Several standard methods run significantly slower and may have less utility as high-throughput sequence analysis for precision immunotherapy becomes more common. Performance of all methods varies by MHC allele, and most of this variance can be explained by data-driven, rather than biological properties. Increasing the quantity of publicly available experimental data has the potential to improve all machine learning methods applied to this problem, and in particular deep learning methods.

Quick Guide to Equations and Assumptions

In silico prediction methods used in cancer research must be rigorously evaluated. The machine learning methods assessed here predict a continuous numeric response value (IC50) for each pMHC-I pair. The methods are considered to be supervised, because they learn to predict the response value from training examples, whose IC50 is assumed to be known. They are expected to make correct predictions on a set of independent, held-out test examples. Misleading and overly optimistic evaluations of supervised machine learning methods are common, in particular when there is an overlap of training and test data, or when training and testing is done on a small set of examples that do not reflect the true diversity of the problem space. The benchmark set used in this work [1] completely separates training and test examples and covers 50 MHC Class I alleles. Performance on this benchmark is a good indicator of the effectiveness of a prediction method. Each predicted response value can be related to an experimental pMHC-I IC50 measurement through the equation $y = max(0, 1 - log_{50K}\text{IC50})$. The equation is used to reduce the possible range of predicted values from between 0 and 50,000nM to between zero and one, a range of values that is more mathematically tractable for neural network optimization algorithms. The assumption is that the largest IC50 value that can be observed is 50,000nM.

There are many ways to evaluate the predictive performance of the output response values. We selected three metrics, one of which measures the correlation between the response values of each predictor and experimental measurements (Kendall-Tau correlation), two which assess each predictor as a classifier of binding vs. non-binding pMHC-I pairs, using an IC50 threshold of 500nM to separate binders and

non-binders (AUC, F1 score).

To compute Kendall-Tau correlation, the response values of a predictor are ranked and compared to a ranked list of experimentally measured pMHC-I IC50 values from the Immune Epitope Database. The Kendall-Tau correlation is then:

$$\tau = \frac{n_c - n_d}{(n_0 - n_1)(n_0 - n_2)} \tag{1}$$

where n_c is the number of concordant pairs (a prediction and observation are concordant if the pMHC-I pair has the same rank in both lists), n_d is the number of discordant pairs. n_0, n_1, and n_2 are used to handle rank ties.

The F1 and AUC scores are also computed by comparing the response values of a predictor with an ordered list of experimentally measured pMHC-I IC50 values. In the case of the F1 score, both lists (the list of predictions and the list of observations) are dichotomized by labeling binders and non-binders. The dichotomization assumes any IC50 $\leq$ 500nM is a binder and a non-binder otherwise. The two lists are then compared and precision is calcluated as the number of items in the list where the prediction of binder agrees with the observation, divided by the number of observed binders. Recall is calaculated as the number of items where the prediction of binder agrees with the observation, divided by the number of items in the list. For AUC, the list of predicted output values is compared to a dichotomized list of observations, and the true positive rate and false positive rate at every threshold on the list is computed. All metrics are described in detail in (Materials and Methods).

Introduction

Peptide binding to Major Histocompatibility Complex (MHC) proteins and formation of a stable binary complex (pMHC) is a key step in the activation of CD8+ (MHC Class I) and CD4+ (MHC Class II) T-cells. MHC Class I proteins bind peptides of 8-11 amino acid residues. The presentation of pMHCs on the surface of antigen-presenting cells and recognition by T-cell receptors is fundamental to the mammalian adaptive immune system. Recent advances in cancer immunotherapy have highlighted the need for improved understanding of which peptides will bind to MHC proteins and generate an immune response [2], [3], [4], [5], [6]. Because experimental characterization of pMHC binding is costly and time-consuming, computational researchers have been working for decades on *in silico* tools to predict pMHC affinities [7]. Approaches have included sequence-based profiles [8] [9], structure-based predictions [10], generative

probabilistic models [11], and machine learning [12] [13] [14].

Neural network supervised machine learning methods are the most widely-used technique and have been shown to outperform other methods in multiple studies [15] [1] [16]. However, many researchers remain dissatisfied with the available *in silico* pMHC binding predictors for neoantigen discovery, particularly in a clinical setting [17] [18]. We explored whether the most recent generation of neural network algorithms and architectures, known as deep learning, could effectively improve pMHC binding predictions.

Neural networks were originally modeled on the human brain. They consist of connected units, organized into layers. Theoretically, given enough layers and units, they can act as universal function approximators [19]. However, as the number of units increases, they become prone to overfitting (reviewed in [20]). Deep learning research has introduced innovative network architectures and regularization techniques, allowing for training of very deep and wide networks to approximate complex functions, with reduced risk of overfitting. Given sufficient training data, deep architectures outperform other methods in many domains [21]. They are now widely used in robotics, self-driving vehicles, sentiment classification, machine translation, and user-based recommendation systems [22] [23] [24] [25] [26].

In this work, we assessed the performance of major deep learning architectures: convolutional neural networks (CNNs) [27] and recurrent neural networks, particularly gated recurrent units (GRUs) [28], and long short-term memory units (LSTMs) [29]. The deep learning methods were compared to several standard machine learning methods: fully-connected one-layer networks (NetMHC [30], NetMHCpan [31], and our own MHCnuggets-FC), a two-layer embedding network (MHCflurry) [32], and a Bayesian least-squares method (SMMPMBEC) [15]. We evaluated the methods on a carefully designed and previously published benchmark set of immuno-fluorescent binding experiments for diverse pMHC-I allele pairs, described hereafter as the *Kim benchmark set* [1], derived from the IEDB database [33]. The Kim benchmark set was cleaned for redundancy, using a protocol designed by [32]. In our hands, the best deep learning methods had similar prediction performance to the best standard machine learning methods. Furthermore, the performance of these methods was very similar both on the aggregate set of all alleles, and when alleles were evaluated individually. Predictions for individual pMHC-I pairs by the best methods were highly correlated (≥ 0.9). Convolutional neural networks, a popular deep learning technique, had poorer performance and do not appear to be well suited for pMHC-I binding prediction (Results). Of the best-performing six methods, three methods developed by us (MHCnuggets-FC, MHCnuggets-LSTM, MHCnuggets-GRU) had the fastest runtimes. In contrast to standard neural networks, which require that

all inputs be of the same length, the deep learning networks MHCnuggets-LSTM and MHCnuggets-GRU can handle peptides of any length. They do not require the artifical lengthening and shortening heuristics (padding and cutting) that are used by MHCnuggets-FC, MHCflurry, NetMHC, and NetMHCpan.

MHC proteins are highly polymorphic, and there are thousands of MHC Class I alleles, each with a different peptide binding surface. While there are tens of thousands of experimentally characterized binding affinities in the Kim benchmark set, they are distributed unevenly across the alleles. For the most common alleles, particularly those associated with European ancestry, there are as many as 9000 characterized peptides, while for others there may be as few as 100. When we stratified prediction performance by individual alleles, performance was seen to vary widely by allele, but not by the *in silico* method.

We explored whether the allele-specific differences in *in silico* binding prediction were data-driven or due to biological properties of the different alleles. Most of the variance in prediction performance could be explained by differences training or test set size, class imbalance, or sequence diversity of peptides in the training set. Because the differences appeared to be primarily data-driven, we designed a training protocol in which information from alleles with the richest training data was shared with other alleles, a form of transfer learning [34]. We clustered the MHC Class I alleles based on the utility of the transferred information between them, yielding improvements in prediction performance for most alleles (Materials and Methods, Results).

Deep learning was originally designed for computational problems where a large amount of training data is available. As data size increases, deep learning performance can increase dramatically, as has been shown in the field of image proccessing, where deep learning methods quickly advanced to be the best in class by a wide margin [35]. It is likely that deep learning methods will benefit the most from increases in experimental training data and will become increasingly relevant to the field of immunology.

Software for the MHCnuggets neural networks described in this work is available at `https://github.com/KarchinLab/mhcnuggets`.

Results

Overview of approach

We developed a new machine learning approach to predict peptide binding to MHC Class I proteins, using deep recurrent neural networks. Our approach differs from previous machine learning methods designed for the same task, which augment their training data artificially or train separate networks for each peptide length. The artificial augmentation used by previous methods includes *in silico* shortening and lengthening of each peptide and generation of non-binding peptides, both of which introduce noise. The recurrent neural networks use only experimentally verified peptide-MHC binding pairs as training data and naturally handle peptides of different lengths. We trained networks for individual MHC alleles and developed a protocol to share information between networks. We hypothesized that this approach could achieve state-of-the art prediction performance at significantly faster speeds. Additionally, we reasoned that these methods could be adapted to improve standard neural networks and to develop a standard network with similar performance and faster runtime.

The best deep and standard machine learning methods have similar prediction performance in aggregate testing of all MHC Class I alleles

We tested several deep learning methods: convolutional neural networks (CNNs) [27], and recurrent neural networks, in particular long short-term memory networks (LSTMs) [29] and gated recurrent units (GRUs) [28]. In total we tested five architectures, four designed by us. We also tested two types of standard machine learning methods: fully connected one- or two-layer networks and a Bayesian matrix method. Five standard methods in total were tested, one designed by us. The best performing deep methods (MHCnuggets-LSTM, MHCnuggets-GRU) and the best standard methods (MHCnuggets-FC, MHCflurry, NetMHC, NetMHCpan) had similar prediction performance. Deep convolutional neural networks (CNNs) had the weakest performance (Table 1).

To the best of our ability, all training and testing of methods was done with the Kim benchmark set. However, for NetMHC and NetMHCpan, open source training software was not available, and we used their self-reproted results on the Kim benchmark set. These methods have been previously reported to augment their training sets with additional proprietary data, beyond what is in the Kim benchmark training set [1, 32], so their self-reported performance estimates may be overly optimistic.

Prediction performance was assessed with three metrics (Materials and Methods). Dichotomous classification of peptides as binders or non-binders was measured with area under the ROC curve (AUC) and F1 scores, with IC50 of 500nM used as the separating threshold [36]. Continuous prediction of IC50 measurements was assessed with the Kendall-Tau correlation coefficient.

The predictions of the best deep and standard machine learning methods are highly correlated

We identified six best-in-class methods (Table 1) and for each method, ranked the predicted IC50 binding affinities for all pMHC-I partners in the Kim benchmark set. The ranked predictions were strikingly correlated (Spearman-Rho rank correlation ≥ 0.9 for all method pairings) (Figure 2). Given that the six methods use different machine learning algorithms, training protocols, and network architectures, the high rank correlation on the level of individual predictions was surprising.

For individual MHC Class I alleles *in silico* prediction performance varies

Next, we compared how the six best performing methods from Table 1 handled each of the MHC Class I alleles. Performance was assessed with the Kendall-Tau correlation. (Figure 3). For a few of these alleles, one of the methods slightly outperformed the others. For example, MHCnuggets-LSTM has slightly higher Kendall-Tau correlations for HLA-A*26:02 and HLA-B*15:09 than other methods. However, in general the alleles could be clustered into two groups, a group of alleles in which all the methods performed well and a group in which none of the methods performed well (dendogram on left hand side of Figure 3). This result is consistent with the high Spearman rank correlation among the methods with respect to predictions of pMHC-I affinities (Figure 2), and their similar performance on the Kim test benchmark set (Table 1).

The best six methods index the columns of the heatmap in Figure 3, and the 50 MHC Class I alleles in the Kim test set index the rows. Each cell represents the performance of a method on an allele's test set, measured by Kendall-Tau correlation. For example, the top right cell shows the Kendall-Tau for MHCnuggets-LSTM and H-2-Kb. Figure S1 shows heatmaps for F1 score and AUC.

	AUC	F1	K-Tau
Deep Networks			
MHCnuggets-GRU	0.931	0.810	0.589
MHCnuggets-LSTM	0.931	0.806	0.587
MHCnuggets-Spanny-CNN	0.918	0.795	0.563
HLA-CNN	0.874	0.750	0.480
MHCnuggets-Chunky-CNN	0.845	0.640	0.447
Standard one- and two-layer networks			
NetMHCpan	0.933	0.803	0.584
MHCflurry	0.933	0.785	0.587
NetMHC	0.932	0.808	0.588
MHCnuggets-FC	0.931	0.814	0.581
Bayesian model			
SMMPMBEC	0.921	0.791	0.578

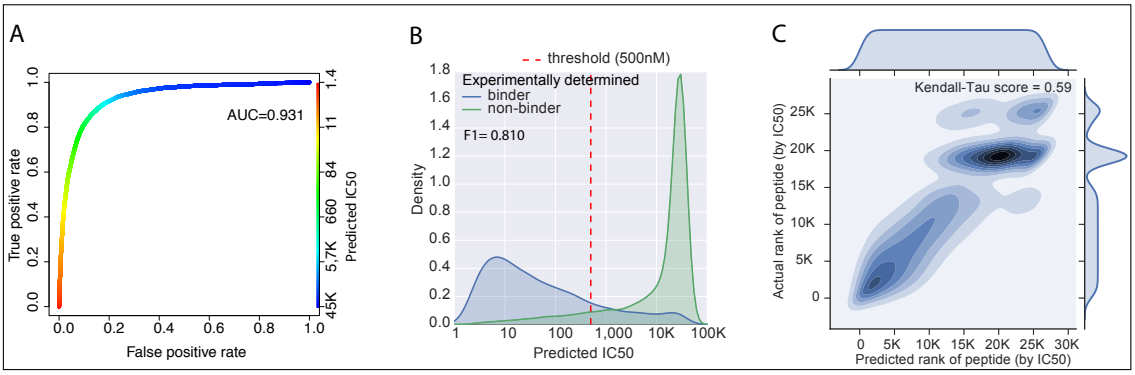

Figure 1: **Methods with best performance for predicting pMHC-I binding.** Table shows performance of ten evaluated machine learning methods. Best performing methods highlighted in gray. Evaluation was done on the Kim benchmark set (50 MHC Class I alleles). Figure shows AUC, F1, and Kendall-Tau performance evaluation on for the gated recurrent unit MHCnuggets-GRU. **A. Area under the receiver operating characteristic curve (AUC)** shows the true positive rate as a function of the false positive rate over a series of IC50 thresholds (color bar). The maximum AUC is 1.0 and random prediction is 0.5. **B. F1 score.** Densities of predicted IC50 scores for binding (blue) and non-binding (green) peptides. Red dotted line shows the threshold of 500nM used to compute the F1 score. The maximum F1 score of 1.0 would completely separate the blue and green densities. F1 scores range from 0.0 to 1.0 **C. Kendall-Tau correlation coefficient** compares two ranked lists based on number of concordant and discordant pairs and handles ties. Y-axis shows the true rank of peptides by IC50 and X-axis shows the predicted rank of peptides by IC50.

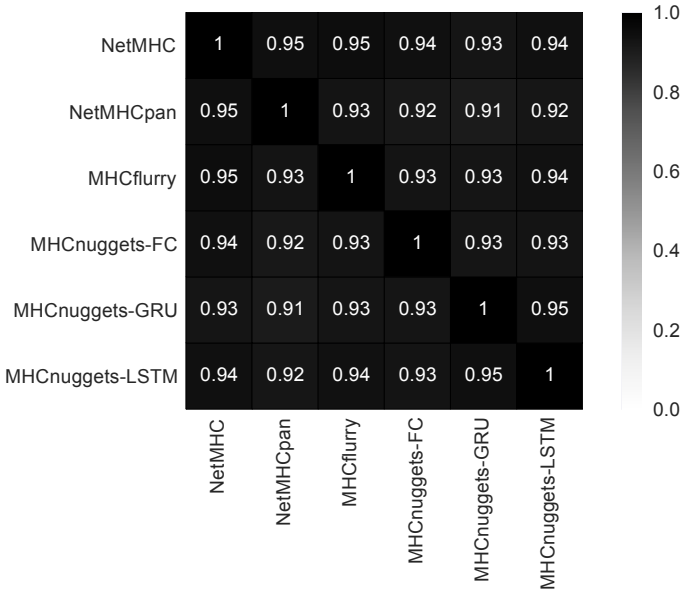

Figure 2: **Rank ordering of peptide binding affinity (pMHC-I IC50) predictions is highly correlated for the six best *in silico* methods.** Spearman's rank order correlations are shown for each pair of methods.

Allele-specific differences in *in silico* prediction performance are largely data-driven

We considered whether properties of the training and test data in the Kim benchmark set could explain the variation in prediction performance for different MHC Class I alleles. For each of the performance metrics, we used ANOVA comparison of nested linear regression models to assess the variance in performance explained by: the number of training examples (peptides with measured binding affinities) for each allele, the class imbalance in binder and non-binder peptides in the training examples, the binder/non-binder class imbalance in the test set, the sequence diversity of the peptides in the training examples (measured by network modularity) (Materials and Methods), the *in silico* method used (only the best six methods were considered), and the number of examples in the test set. Choice of *in silico* method and test set size did not explain any additional variance in performance, once the most informative covariates were considered. For performance measured by Kendall-Tau correlation, 70.4% of the variation across alleles was explained by test set class imbalance (65.5%, p=5e-11), training set size (4.1%, p=5e-11), and sequence diversity of all peptides in the training set (0.5%, p=0.01). For performance measured by F1 score, 56.4%

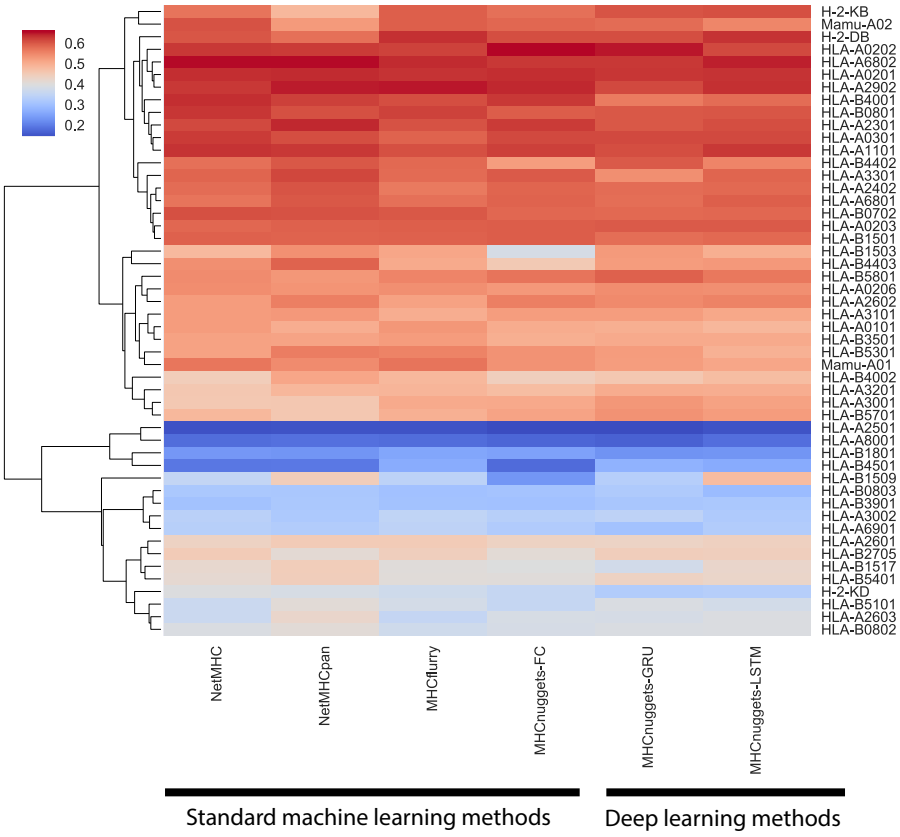

Figure 3: ***In silico* prediction performance of the six best methods varies considerably by individual MHC Class I allele but does not vary much by prediction method.** In the hierarchically clustered heat map, the methods index the columns and alleles index the rows. Heat map coloring indicates high (red) vs. low (blue) Kendall-Tau correlations. The high variability by allele is seen by looking up or down any column. The low variability by method is seen by looking across any row. The alleles cluster into two groups, one in which all of the methods perform well and one in which none of the methods perform well (upper and lower clades of dendogram on left-hand-side).

of the variation across was alleles was explained by training set class imbalance (35.3%, p=1e-34), training set size (9.8%, p=4e-14), test set class imbalance (3.8%, p=7e-07), binder diversity (3.9%, p=2e-07), all diversity (3.6%, p=1.62e-07). For performance measured by AUC, only 25% of the variation was explained by test set class imbalance (12%, p=1e-11) and training set size (12.7%, p=2e-13), but we believe that this is because the AUC is the least stringent of the three metrics. Our results suggest that properties of the training and test data explain a large proportion of the variance among performance of *in silico* methods across different MHC Class I alleles, and that biochemical properties of allele-specific pMHC-I complexes are not the main contributor.

Transfer learning improves prediction performance

Transfer learning is a technique commonly used in deep learning, in which information is transfered between distinct but related problem domains [34]. We designed a new transfer learning protocol for neural networks trained on individual MHC alleles (Figure 4), and applied it to the best performing networks (MHCnuggets-GRU, MHCnuggets-LSTM, MHCnuggets-FC). The protocol was compared to standard random initialization of network weights with the Glorot uniform distribution [37]. Transfer learning improved prediction performance for most of the alleles, and for several alleles which had the worst *in silico* prediction performance across all tested methods, the improvement was substantial (*e.g.,* HLA-B*08:02, HLA-B*08:03, HLA-A*25:01, HLA-B*15:17) (Figure 5). The relationships between alleles defined by utility for transfer learning had overlap to standard MHC nomenclature and to MHC allele organization into supertypes [38] [39]. The performance differences measured by Kendall-Tau correlation and AUC are in Figure S2.

Runtime of the best deep and standard machine learning methods varies substantially

We computed the runtime to score all the peptides in the Kim benchmark set (163,898) for a single MHC Class I allele, for the six best perfoming methods. Computations were repeated five times and averaged. The fastest method was MHCnuggets-FC (7.32s±0.05) and the slowest method was NetMHCpan (282.58s±18.64). The best-performing deep learning MHCnuggets methods were relatively fast (22.76s±0.5 for MHCnuggets-GRU, 23.91s±0.37 for MHCnuggets-LSTM). Timing was done on a single compute node

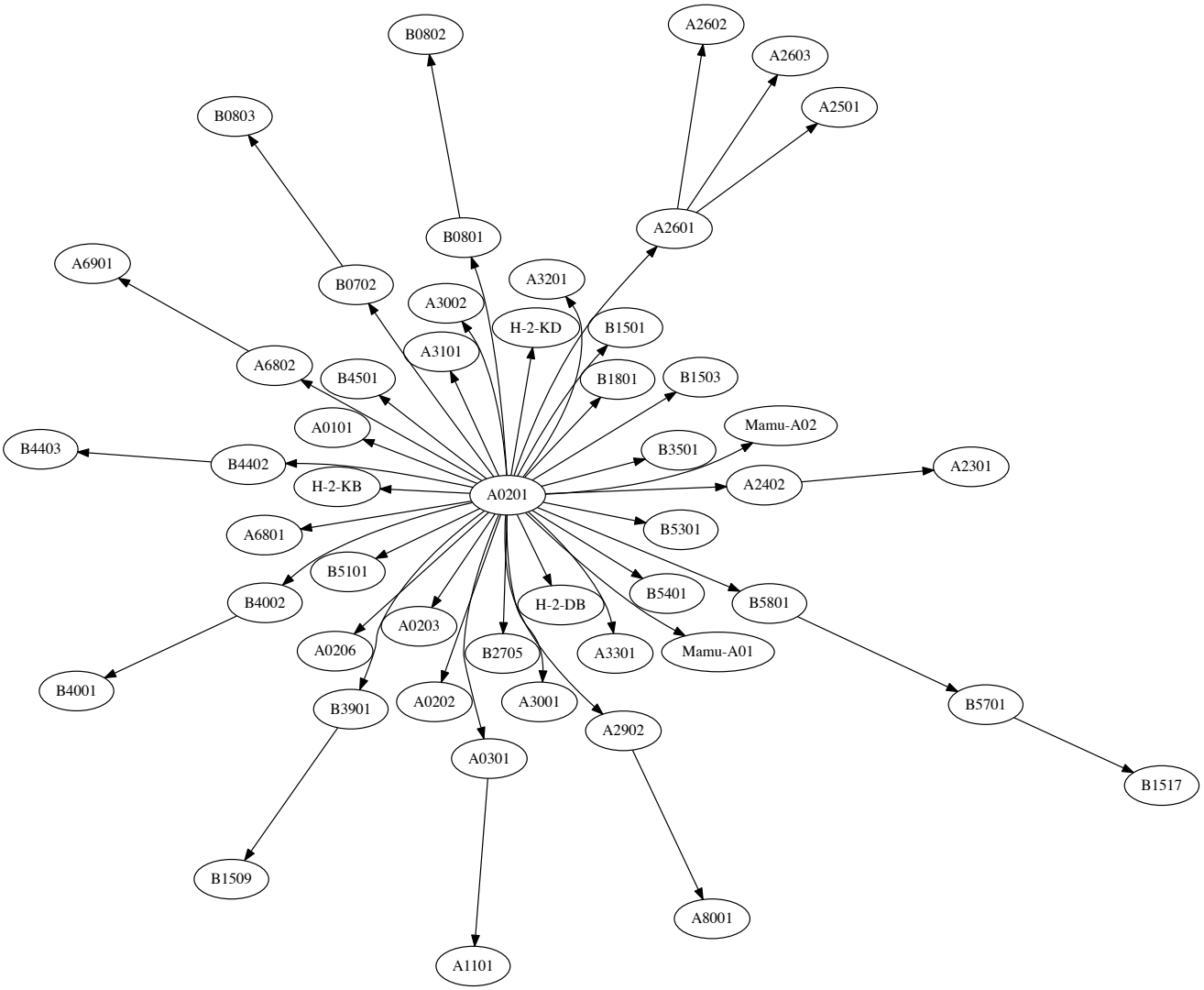

Figure 4: **Transfer learning directed graph.** The graph shows transfer of information between MHC Class I alleles in our network training protocol. Nodes represent MHC class I allele-specific neural networks and directed edges show how trained network weights are transfered from one network to another. The first network trained is for HLA-A*02:01 binding peptides, and the trained weights are used to initalize the networks for all other alleles, yielding 50 additional allele-specific networks. These networks are tested for their prediction performance on the training data for each of the 50 alleles. The network weights for the best performing networks are transferred to the corresponding alleles. This three-step protocol is shown in Figure S2.

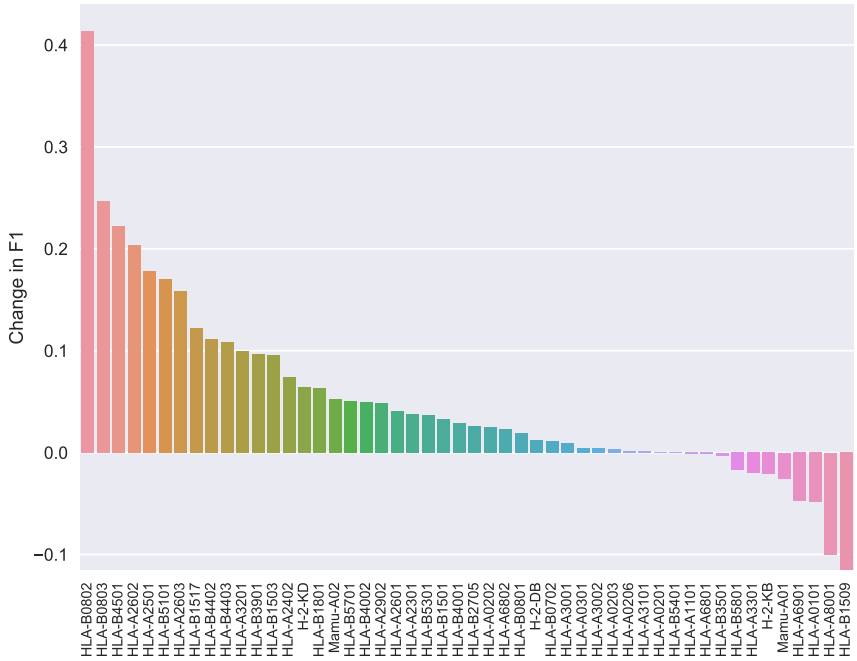

Figure 5: **Transfer learning improves overall performance for deep learning networks.** The difference in F1 score between deep learning MHCnuggets-GRU networks trained with and without the transfer learning protocol (Figure 4). The 50 MHC class I alleles in the Kim benchmark set are shown and the bar height and color represent the net change in F1 score after transfer learning was applied. The F1 score improved for most alleles, and substantial decrease in score was seen for only two alleles.

Method	Kim benchmark runtime [s]	Hypothetical cohort runtime [min]
MHCnuggets-FC	7.32±0.05	9.12
MHCnuggets-GRU	22.76±0.50	28.38
MHCnuggets-LSTM	23.91±0.37	29.81
MHCflurry	49.12±1.33	61.24
NetMHC	80.89±10.3	100.85
NetMHCpan	282.58±18.64	352.29

Table 1: **Runtime analysis of six best performing methods (fastest to slowest)**. For each method, the average runtime for predicting pMHC-I affinity of all peptides (163,898) in the Kim benchmark for a single HLA allele is shown. Estimates of the runtime for each predictor on the somatic missense mutations in a cancer cohort of 495 patients and 4128 candidate neoantigen peptides per patient are shown.

containing 2 Intel Xeon E5-2680v3 (Haswell) processors running at 2.5GHz, with 12 cores and 128 GB RAM.

The runtime differences are substantial if we estimate the time to run each predictor on a cohort of cancer patients, based on their somatic missense mutations. For example, the head and neck cancer (HNSC) cohort from TCGA has 495 samples (as of June 15, 2017), with six potential MHC Class I alleles per patient (3 loci, possibly all heterozygous). If we consider all possible 8-11 length windows around each somatic missense mutation, the average candidate peptides per patient is 4128. For MHCnuggets-FC, the run would take approximately 9 minutes, for MHCnuggets-GRU and MHC-nuggets-LSTM approximately 30 minutes, and for NetMHCpan approximately 6 hours. Complete timing results are shown in Table S1.

Discussion

Recent advances in precision cancer immunotherapy have highlighted the need for improved *in silico* methods to predict peptide-MHC (pMHC) binding affinities. A patient's repertoire of somatic mutations may yield processed peptides, which are presented to T-cell receptors by MHC proteins to trigger an immune response. Therefore, high-throughput *in silico* prediction of which somatic mutations are most likely to generate immunogenic peptides has increasing clinical utility. These mutations may serve as biomarkers to identify patients most likely to respond to checkpoint blockade and may also have utility for engineered vaccines, designed to stimulate immune response and destroy cancer cells. [17] [18]. As new *in silico* methods are developed, it is critical that they are carefully assessed to determine whether new algorithms, such as deep learning networks, provide advantages over established ones. In this work, we focused on the Kim benchmark set [1], which contains a sizeable and heterogeneous population of

experimentally characterized peptides and pMHC-I affinities, cleanly separated into training and test data. By comparing ten methods with the Kim benchmark, we avoided bias that could be introduced by smaller or less diverse benchmark sets. Of the ten methods tested, we found that six were best-in-class with respect to prediction performance. Three of the methods, which were designed by us, also had signficantly faster run-times, which may be important to users who are interested in scalable, high-throughput neoantigen prediction pipelines.

Because MHC alleles are highly polymorphic, particularly at the peptide binding site, each allele has its own repertoire of binding peptides. Experimental tests of pMHC-I binding have not been uniformly applied to all alleles, with certain alleles such as HLA-A*02:01 represented with 12,357 experiments in the Immune Epitope Database (IEDB) [33] and 9,565 in the Kim benchmark. Other alleles such as HLA-B*08:03 have very few experimentally validated pMHC-I affinities (470 in IEDB, 217 in the Kim benchmark). Furthermore, the class imbalance between experimentally tested binders and non-binders ranges from 0.041 in HLA-B*08:02 (19 binders, 468 non-binders) to 0.86 Mamu-A*02 (1003 binders, 1261 non-binders). *In silico* predictors work better for some MHC class I alleles than others. It has been hypothesized that there is a biological/chemical/structural explanation, so that an immune response might be generated from weaker affinity peptide binding for some alleles, but only stronger affinity for others. If this is true, 500nM may not be a good separator of binding and non-binding peptides for all alleles, and allele-specific affinity thresholds should be considered [40]. While we cannot rule out the relevance of biologically-driven factors, we found that the most of the variance in prediction performance across the different alleles could be explained by several data-driven factors. We designed a linear regression in which the response variable was a performance metric (Kendall-Tau correlation, F1, or AUC) as calculated for the best six methods, and the covariates were values that differed across the 50 alleles in the Kim benchmark. Choice of *in silico* method was also considered as a covariate. For Kendall-Tau correlation, test set class imbalance, training set size, and sequence diversity of training set peptides explained 70.4% of the variation. For F1 score and AUC, covariates explained 56.4% of the variation and 25% of the variation, respectively. Choice of *in silico* method did not explain any additional variance in performance for any of the metrics, once the most informative covariates were considered. Our results suggest that properties of the training and test data explain a large proportion of the variance among performance of *in silico* methods across different MHC Class I alleles, and that biochemical properties of allele-specific pMHC-I complexes are not the main contributor.

When we applied the machine learning technique of transfer learning to MHCnuggets-GRU, MHCnuggets-LSTM, and MHCnuggets-FC, we observed a substantial increase in performance, in particular the F1 score and Kendall Tau coefficient. Our transfer learning approach leveraged a network of related MHC Class I alleles. Base networks were trained on alleles with a larger number of training examples, and trained weights were transferred to initialize the network of a related allele. Our results indicate that transfer learning is a useful technique for pMHC-I binding prediction, in particular for alleles with few training examples. The relationships between alleles that we found empirically were in agreement with previously described MHC supertypes, but also included cases where more distant relationships provided useful transferable information from one allele's training data to another.

Of the deep learning methods we investigated, CNNs were the least successful. We attribute this result to the importance of positional information in the pMHC-I binding process. It is well known that specific residues are required at anchor positions [41] The filter sizes typically used in CNNs were designed for problems in which position invariance is critical, such as identifying an object regardless of its position or angle within an image. In contrast, the presence of a leucine amino acid residue has different meaning at a pMHC-I anchor position than elsewhere, thus position invariance is detrimental. We were able to improve CNN performance by increasing filter size, but they were still unable to match the performance of the GRU, LSTM, or standard neural networks.

Our work highlights the critical role of experimental training data for further development of *in silico* pMHC binding predictors. Even the best methods identified by our benchmarking efforts have sharply decreased performance for those MHC-I alleles having a small and/or highly imbalanced number of experimental measurements, and most of the variance in predictive performance could be explained by these data-driven factors. We encourage anyone who is measuring pMHC binding affinities through immuno-fluorescent assays or other technologies to deposit their results in public databases such as IEDB.

Materials and Methods

Dataset

The Immune Epitope Database (IEDB) [33] provides a comprehensive public set of experimentally characterized peptides and pMHC binding affinities. Database entries were curated from published literature, and the majority of affinities were calculated based on immunofluorescent assays. These

affinities are represented as an IC50 value, the half-maximal inhibitory concentration in nano-molar (nM) units of peptide to MHC molecules. A total of approximately 250,000 examples from immunofluorescent assays was available as of May 2017, spanning multiple mammalian and avian species, and 740 MHC alleles, of which 459 are MHC Class I.

For purposes of benchmarking *in silico* pMHC Class I binding predictors, Kim et al. generated a new dataset from IEDB, partitioned into training and test sets [1]. The dataset was further processed by a shell script [32] and available at `https://github.com/hammerlab/mhcflurry/tree/master/downloads-generation/data_kim2014`, that removed any peptide in the test set with identical length and $\geq 80\%$ sequence identity to a peptide in the training set. The resulting benchmark is referred to in this work as the Kim benchmark. It contains 106 unique MHC alleles and 137,654 IC50 measurements, published prior to 2009 (training set) and 53 unique MHC alleles with 27,680 IC50 measurements, published from 2009-2013 (test set). All peptides in the Kim benchmark set consist of 8-11 amino acid residues.

In silico methods

MHCnuggets

Recurrent neural networks

MHCnuggets-GRU is a gated recurrent unit (GRU) [28] with a fully connected layer of 64 hidden units, regularized with a dropout and recurrent dropout [42] probability of 0.2, trained for 200 epochs. MHCnuggets-LSTM is a long short-term memory network [29] with a fully connected layer of 64 hidden units, regularized with a dropout and recurrent dropout probability of 0.2, trained for 100 epochs. Both accept variable length inputs.

Convolutional neural networks

MHCnuggets-Chunky-CNN and MHCnuggets-Spanny-CNN are 1D convolutional neural networks [27]. The Chunky-CNN fits network weights to sliding windows (kernels) across each peptide, which cover two or three amino acid residues. The Spanny-CNN has an additional kernel that covers the entire peptide. Both networks have a single fully connected layer, with 64 hidden units. These networks require fixed length inputs.

MHCnuggets-FC

MHCnuggets-FC is a fully-connected single-layer network with 64 hidden units, regularized with a dropout probability of 0.8, trained for 250 epochs. This network requires fixed length inputs, but we designed a simplified cutting and padding procedure to minimize noise and computation time.

For all MHCnuggets architectures, a separate network was trained for each MHC allele in the Kim benchmark set. Additional technical details are provided in Supplementary Materials and Methods.

Transfer learning

We designed a transfer learning protocol that leverages relationships between MHC alleles, whose peptide binding surfaces may have common biochemical properties. We reasoned that if such a similarity existed, the final learned weights for one MHC allele's network could contain useful information for other alleles. In the transfer learning protocol, the final weights from one network are used as the initial weights for training subsequent networks. To identify potential similarities, we applied an empirical, bottom-up approach (Figure S3). First, we extracted the network weights from the MHC allele with the largest number of training examples (HLA-A*02:01). The trained weights from this network were used to intialize and train networks for the remaining 50 alleles. For some alleles, HLA-A*02:01 was not expected to have the most transferable information, so we applied all of the trained networks to score the training examples for all alleles. In many cases, there was an allele whose network had better predictive performance than HLA-A*02:01 at this step. For these alleles, we extracted the trained weights from the best performing network and used them to initialize a new round of network training. Figure 4 shows the final transfer learning protocol, with nodes representing the networks for each allele and edges represent the direction and transfer of weights from one network to another. We used AUC to measure network performance (all three performance metrics were highly correlated) and required that to be utilized for transfer learning, a source network had a larger number of training examples than its target and a perfomance AUC ≥ 0.9. To avoid overfitting, no examples from the Kim benchmark test set were used.

NetMHC/NetMHCpan

NetMHC [30] is a single-layer fully connected neural network that encodes nine amino acid residue peptides with a 378-length input vector, which incorporates both a smoothed one-hot encoding (0.9 and 0.05

replace 1 and 0) and a BLOSUM-62 encoding of each amino acid. To accommodate peptides that are shorter or longer than nine residues, contiguous padding or cutting operations are applied at every possible position. When padded or cut versions of the same peptide receive different predicted binding affinities, the strongest affinity is selected. Separate networks are trained for each MHC allele.

NetMHCpan [31] is also a single-layer fully connected neural network that encodes both peptide and polymorphic residues of the relevant MHC allele. It uses the same smoothed one-hot plus BLOSUM-62 encoding and padding/cutting protocol as NetMHC. A single network is trained for all MHC alleles. NetMHC and NetMHCpan are currently the most widely used in silico pMHC-I binding tools.

Both methods use artificially generated non-binding peptides, by applying the NetChop algorithm [43] to the entire human proteome.

MHCflurry

MHCflurry [32] is a neural network, which jointly discovers informative amino-acid residue encodings and predicts pMHC-I binding affinities. Each amino acid residue type is assigned an integer, and peptides are encoded as 9-length vectors of integers. An initial embedding layer maps input peptides to a 32-dimensional space, which then feeds into a fully connected layer. Padding and cutting of peptides that are shorter or longer than 9 residues is done identically to the protocol of NetMHC/NetMHCpan. The final predicted binding affinity is the geometric mean of all padded and cut versions of the same peptide. MHCflurry augments its training data with peptides randomly generated *in silico* from a uniform distribution.

SMMPMBEC

SMMPMBEC [15] is a Bayesian framework for regularized least-squares regression. Peptides to be used for training are represented with one-hot encoding and stacked to form a single NxL matrix, where N is the number of peptides and L is the peptides' length x 20. A scoring matrix is trained to minimize the error between N experimental binding affinities and the matrix product of the peptide matrix and the scoring matrix. The optimization is constrained by a pre-trained 20x20 pairwise substitution matrix for the amino acids, based on the covariance of their contributions to pMHC-I binding free energy in different contexts. A scoring matrix is computed for each MHC allele and each peptide length. To predict the pMHC-I binding affinity for a peptide of interest, its one-hot representation is multiplied by the scoring matrix.

HLA-CNN

HLA-CNN [13] is a deep learning convolutional neural network, consisting of an embedding layer, two 1D convolutional layers, and a fully connected layer. The input to the embedding layer is an Lx15 matrix, which is initialized with a learned peptide representation based on the natural language processing (NLP) skip-gram model [44]. Skip-gram is an unsupervised technique to discover word embeddings. Sentences are encoded as vectors of integers and projected into a lower dimensional space. HLA-CNN treats peptides as sentences and amino acid residues as words. The initial embedding is learned from the set of all peptides across all MHC alleles in the training set. The convolutional layers consist of 32 filters, with kernel size of 7, stride=1, and are initialized with Glorot normal distribution [37]. A network is trained for each MHC allele and each peptide length.

Comparison metrics

All *in silico* pMHC-I binding predictors assessed in this work are fundamentally regression methods, which produce a continuous-valued prediction, expected to be related to the true experimental IC50 measurement. Continuous-valued predictions can be thresholded at relevant biological IC50s to produce dichotomous class predictions (binder or non-binder). We used an IC50 threshold of 500nM, where a value $\leq$ 500nM implied a binder and $>$ 500nM implied a non-binder [36]. We applied the Kendall-Tau correlation coefficient for continuous-valued predictions (Equation 2)

$$\tau = \frac{n_c - n_d}{(n_0 - n_1)(n_0 - n_2)} \tag{2}$$

where n_c is the number of concordant pairs, n_d is the number of discordant pairs, and n_0, n_1, and n_2 are given by Equations 3, 4, 5.

$$n_0 = n(n-1)/2 \tag{3}$$

$$n_1 = \sum_i t_i(t_i - 1)/2 \tag{4}$$

$$n_2 = \sum_j u_j(u_j - 1)/2 \tag{5}$$

where n is the total number of experimental and predicted binding affinity pairs, t_i is the number of tied values in the i^{th} group of tied experimental affinities, and u_j is the number of tied values in the j^{th} group of tied predicted affinities.

The Kendall-Tau coefficient provides assessment of whether continous predictions and experimental measurements have a monotonic relationship and is robust to non-linearity.

We used area under the ROC curve (AUC) and F1 (Equation 6) metrics to assess dichotomous class predictions. AUC assesses the dichotomous classification between binders and non-binders at any threshold of a predictor's continuous output. At each threshold, the true positive rate (precision) and false positive rate ($\frac{\text{number of non-binders predicted as binder}}{\text{number of true non-binders}}$) is computed and a curve is plotted (Figure 1A). The final metric is the area under this curve.

$$F1 = \frac{2 * (precision * recall)}{(precision + recall)} \tag{6}$$

where precision $= \frac{\text{number of correctly predicted binders}}{\text{total true binders}}$ and recall $= \frac{\text{number of correctly predicted binders}}{\text{total peptides scored}}$. Binders are defined as peptides with IC50 $\leq$ 500nM.

A good AUC does not guarantee that the separation occurs at a biologically relevant threshold. In particular, if the output does not have a good inverse mapping to experimental measurements, the best separation may occur at an uninformative threshold. The F1 score ensures that a dichotomous classification occurs at a selected threshold, and the threshold can be selected to be biologically relevant. It provides a balanced summary of the precision and recall of a dichotomous classification.

Variability in prediction performance

For each of the 50 MHC Class I alleles in the Kim benchmark test set, we calculated the following covariates: the number of training examples n, the training example class imbalance $\frac{n_b}{n_{nb}}$ where n_b is the number of training examples $\leq$ 500nM and n_{nb} is the number $>$ 500nM (in practice we use $abs(ln(\frac{n_b}{n_{nb}}))$ to handle cases where $n_b > n_{nb}$), the test example class imbalance, the network modularity of binding peptides in the training set, and the network modularity of all peptides in the training set. The ANOVA R package was utilized to measure the variance explained by each covariate, using nested linear regression models and with either F1 score, Kendall-Tau score, or AUC as the response variable. covariates were added to each model with a greedy maximization of variance algorithm, in which the covariate that

increased the variance the most at each iteration was added to the model, if it had a p-value < 0.05.

Network modularity of peptide sequence analysis

We constructed a network for each of the 50 MHC Class I alleles in the Kim benchmark test set. The nodes of these networks were the peptide sequences found in the training set of each allele. Two nodes were connected by an edge if the sequence identity between the peptides was > 0 (*i.e.*, they shared at least one identical amino acid residue at the same position). Edges were weighted by sequence identity (normalized by peptide length). Peptides were trivially aligned by cutting and padding them to length nine, as described in (MHCnuggets). Community detection was performed with the fast unfolding algorithm [45], as implemented by the `community_multilevel` function in *igraph*. Once nodes were assigned to communities, the weighted modularity of the community assignment was calculated as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{7}$$

where A_{ij} is the weight of the edge between i and j, $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges associated with vertex i, c_i is the community to which vertex i is assigned, δ is an indicator function such that $\delta(u, v) = 1$ if $u = v$ and is 0 otherwise, and $m = \frac{1}{2} \sum_{ij} A_{ij}$.

Acknowledgments

We would like to thank the MHCflurry team for their inspiration and the obvious influence on the name of our methods, MHCnuggets. MHCflurry open sources all of their software and data, including scripts for pre- and post-processing that we used in this work.

References

1. Yohan Kim, John Sidney, Søren Buus, Alessandro Sette, Morten Nielsen, and Bjoern Peters. Dataset size and composition impact the reliability of performance benchmarks for peptide-mhc binding predictions. *BMC Bioinformatics*, 15(1):241, 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-241. URL `http://dx.doi.org/10.1186/1471-2105-15-241`.

2. Giorgio Parmiani, Chiara Castelli, Piero Dalerba, Roberta Mortarini, Licia Rivoltini, France sco M. Marincola, and Andrea Anichini. Cancer immunotherapy with peptide-based vaccines: What have we achieved? where are we going? *JNCI: Journal of the National Cancer Institute*, 94(11):805, 2002. doi: 10.1093/jnci/94.11.805. URL +http://dx.doi.org/10.1093/jnci/94.11.805.

3. Yong-Chen Lu and Paul F. Robbins. Cancer immunotherapy targeting neoantigens. *Seminars in Immunology*, 28(1):22 – 27, 2016. ISSN 1044-5323. doi: https://doi.org/10.1016/j.smim.2015.11.002. URL http://www.sciencedirect.com/science/article/pii/S1044532315000731. T cell therapies for cancer.

4. Rong-Fu Wang and Helen Y. Wang. Immune targets and neoantigens for cancer immunotherapy and precision medicine. *Cell Res*, 27(1):11–37, Jan 2017. ISSN 1001-0602. URL http://dx.doi.org/10.1038/cr.2016.155. Review.

5. Ellis L. Reinherz. tcr-mediated recognition: Relevance to tumor-antigen discovery and cancer immunotherapy. *Cancer Immunology Research*, 3(4):305–312, 2015. ISSN 2326-6066. doi: 10.1158/2326-6066.CIR-15-0042. URL http://cancerimmunolres.aacrjournals.org/content/3/4/305.

6. Valsamo Anagnostou, Kellie N. Smith, Patrick M. Forde, Noushin Niknafs, Rohit Bhattacharya, James White, Theresa Zhang, Vilmos Adleff, Jillian Phallen, Neha Wali, Carolyn Hruban, Violeta B. Guthrie, Kristen Rõdgers, Jarushka Naidoo, Hyunseok Kang, William Sharfman, Christos Georgiades, Franco Verde, Pëter Illei, Qing Kay Li, Edward Gabrielson, Malcolm V. Brock, Cynthia A. Zahnow, Stephen B. Baylin, Robert B̃. Scharpf, Julie R. Brahmer, Rachel Karchin, Drew M. Pardoll, and Victor E. Velculescu. Evolution of neoantigen landscape during immune checkpoint blockade in non–small cell lung cancer. *Cancer Discovery*, 2017. ISSN 2159-8274. doi: 10.1158/2159-8290.CD-16-0828. URL http://cancerdiscovery.aacrjournals.org/content/early/2017/02/15/2159-8290.CD-16-0828.

7. Claus Lundegaard, Ole Lund, Sren Buus, and Morten Nielsen. Major histocompatibility complex class i binding predictions as a tool in epitope discovery. *Immunology*, 130(3):309–318, 2010. ISSN 1365-2567. doi: 10.1111/j.1365-2567.2010.03300.x. URL http://dx.doi.org/10.1111/j.1365-2567.2010.03300.x.

8. Pedro A. Reche and Ellis L. Reinherz. *Prediction of Peptide-MHC Binding Using Profiles*, pages 185–200. Humana Press, Totowa, NJ, 2007. ISBN 978-1-60327-118-9. doi: 10.1007/978-1-60327-118-9_13. URL http://dx.doi.org/10.1007/978-1-60327-118-9_13.

9. Hao Zhang, Ole Lund, and Morten Nielsen. The pickpocket method for predicting binding specificities for receptors based on receptor pocket similarities: application to mhc-peptide binding. *Bioinformatics*, 25(10):1293, 2009. doi: 10.1093/bioinformatics/btp137. URL +http://dx.doi.org/10.1093/bioinformatics/btp137.

10. Chen Yanover and Philip Bradley. Large-scale characterization of peptide-mhc binding landscapes with structural simulations. *Proceedings of the National Academy of Sciences*, 108(17):6981–6986, 2011. doi: 10.1073/pnas.1018165108. URL http://www.pnas.org/content/108/17/6981.abstract.

11. Hideki Noguchi, Ryuji Kato, Taizo Hanai, Yukari Matsubara, Hiroyuki Honda, Vladimir Brusic, and Takeshi Kobay ashi. Hidden markov model-based prediction of antigenic peptides that interact with mhc class ii molecules. *Journal of Bioscience and Bioengineering*, 94(3):264 – 270, 2002. ISSN 1389-1723. doi: http://dx.doi.org/10.1016/S1389-1723(02)80160-8. URL http://www.sciencedirect.com/science/article/pii/S1389172302801608.

12. Pierre Dönnes and Arne Elofsson. Prediction of mhc class i binding peptides, using svmhc. *BMC Bioinformatics*, 3(1):25, 2002. ISSN 1471-2105. doi: 10.1186/1471-2105-3-25. URL http://dx.doi.org/10.1186/1471-2105-3-25.

13. Yeeleng Vang and Xiaohui Xie. HLA class I binding predictions via convolutional neural networks. *Bioinformatics*, 2017. doi: 10.1093/bioinformatics/btx264. URL https://doi.org/10.1093/bioinformatics/btx264.

14. Pavel P. Kuksa, Martin Renqiang Min, Rishabh Dugar, and Mark Gerstein. High-order neural networks and kernel methods for peptide-mhc binding prediction. *Bioinformatics*, 31(22):3600, 2015. doi: 10.1093/bioinformatics/btv371. URL +http://dx.doi.org/10.1093/bioinformatics/btv371.

15. Y. Kim, J. Sidney, C. Pinilla, A. Sette, and B. Peters. Derivation of an amino acid similarity matrix

for peptide: Mhc binding and its application as a bayesian prior. *BMC Bioinforma.*, 10, 2009. doi: 10.1186/1471-2105-10-394. URL http://dx.doi.org/10.1186/1471-2105-10-394.

16. Thomas Trolle, Imir G. Metushi, Jason A. Greenbaum, Yohan Kim, John Sidney, Ole Lund, Alessañdro Sette, Bjoern Peters, and Morten Nielsen. Automated benchmarking of peptide-mhc class i binding predictions. *Bioinformatics*, 31(13):2174, 2015. doi: 10.1093/bioinformatics/btv123. URL +http://dx.doi.org/10.1093/bioinformatics/btv123.

17. David Gfeller, Michal Bassani-Sternberg, Julien Schmidt, and Immanuel F. Luescher. Current tools for predicting cancer-specific t cell immunity. *OncoImmunology*, 5(7):e1177691, 2016. doi: 10.1080/2162402X.2016.1177691. URL http://dx.doi.org/10.1080/2162402X.2016.1177691.

18. X. Shirley Liu and Elaine R. Mardis. Applications of immunogenomics to cancer. *Cell*, 168 (4):600–612, 2017/06/13 2017. ISSN 0092-8674. doi: 10.1016/j.cell.2017.01.014. URL http://dx.doi.org/10.1016/j.cell.2017.01.014.

19. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

20. Douglas M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004. doi: 10.1021/ci0342472. URL http://dx.doi.org/10.1021/ci0342472. PMID: 14741005.

21. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature14539. Insight.

22. Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015. URL http://arxiv.org/abs/1504.00702.

23. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

24. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

25. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

26. Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf`.

27. Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`.

28. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

29. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

30. Claus Lundegaard, Kasper Lamberth, Mikkel Harndahl, Sren Buus, Ole Lund, and Morten Nielsen. Netmhc-3.0: accurate web accessible predictions of human, mouse and monkey mhc class i affinities for peptides of length 8-11. *Nucleic Acids Research*, 36(Web-Server-Issue):509–512, 2008. URL `http://dblp.uni-trier.de/db/journals/nar/nar36.html#LundegaardLHBLN08`.

31. Morten Nielsen, Claus Lundegaard, Thomas Blicher, Kasper Lamberth, Mikkel Harndahl, Sune Justesen, Gustav Rder, Bjoern Peters, Alessandro Sette, Ole Lund, and Sren Buus. Netmhcpan, a method for quantitative predictions of peptide binding to any hla-a and -b locus protein

of known sequence. *PLOS ONE*, 2(8):1–10, 08 2007. doi: 10.1371/journal.pone.0000796. URL https://doi.org/10.1371/journal.pone.0000796.

32. Alex Rubinsteyn, Timothy O'Donnell, Nandita Damaraju, and Jeffrey Hammerbacher. Predicting peptide-mhc binding affinities with imputed training data. *bioRxiv*, 2016. doi: 10.1101/054775. URL http://biorxiv.org/content/early/2016/05/22/054775.

33. Randi Vita, James A. Overton, Jason A. Greenbaum, Julia Ponomarenko, Jason D. Clark, Jason R. Cantrell, Daniel K. Wheeler, Joseph L. Gabbard, Deborah Hix, Alessandro Sette, and Bjoern Peters. The immune epitope database (iedb) 3.0. *Nucleic Acids Research*, 43(D1):D405, 2015. doi: 10.1093/nar/gku938. URL +http://dx.doi.org/10.1093/nar/gku938.

34. Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969197.

35. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

36. A Sette, A Vitiello, B Reherman, P Fowler, R Nayersina, W M Kast, C J Melief, C̃ Oseroff, L Yuan, J Ruppert, J Sidney, M F del Guercio, S Southwood, R T Kubo, R W Chesnut, H M Grey, and F V Chišari. The relationship between class i binding affinity and immunogenicity of potential cytotoxic t cell epitopes. *The Journal of Immunology*, 153(12):5586–5592, 1994. ISSN 0022-1767. URL http://www.jimmunol.org/content/153/12/5586.

37. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

38. Pedro A. Reche and Ellis L. Reinherz. *Definition of MHC Supertypes Through Clustering of MHC Peptide-Binding Repertoires*, pages 163–173. Humana Press, Totowa, NJ, 2007. ISBN

978-1-60327-118-9. doi: 10.1007/978-1-60327-118-9_11. URL http://dx.doi.org/10.1007/978-1-60327-118-9_11.

39. John Sidney, Bjoern Peters, Nicole Frahm, Christian Brander, and Alessandro Sette. Hla class i supertypes: a revised and updated classification. *BMC Immunology*, 9(1):1, 2008. ISSN 1471-2172. doi: 10.1186/1471-2172-9-1. URL http://dx.doi.org/10.1186/1471-2172-9-1.

40. Sinu Paul, Daniela Weiskopf, Michael A Angelo, John Sidney, Bjoern Peters, and Alessandro Sette. Hla class i alleles are associated with peptide-binding repertoires of different size, affinity, and immunogenicity. *Journal of immunology (Baltimore, Md. : 1950)*, 191:5831–5839, December 2013. ISSN 1550-6606. doi: 10.4049/jimmunol.1302101.

41. Y Saito, P A Peterson, and M Matsumura. Quantitation of peptide anchor residue contributions to class i major histocompatibility complex molecule binding. *Journal of Biological Chemistry*, 268 (28):21309–17, 1993. URL http://www.jbc.org/content/268/28/21309.abstract.

42. Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287*, 2015.

43. Can Kemir, Alexander K Nussbaum, Hansjrg Schild, Vincent Detours, and Sren Brunak. Prediction of proteasome cleavage motifs by neural networks. *Protein engineering*, 15:287–296, April 2002. ISSN 0269-2139.

44. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

45. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

46. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

47. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2627435.2670313`.

48. François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

49. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `http://arxiv.org/abs/1603.04467`.

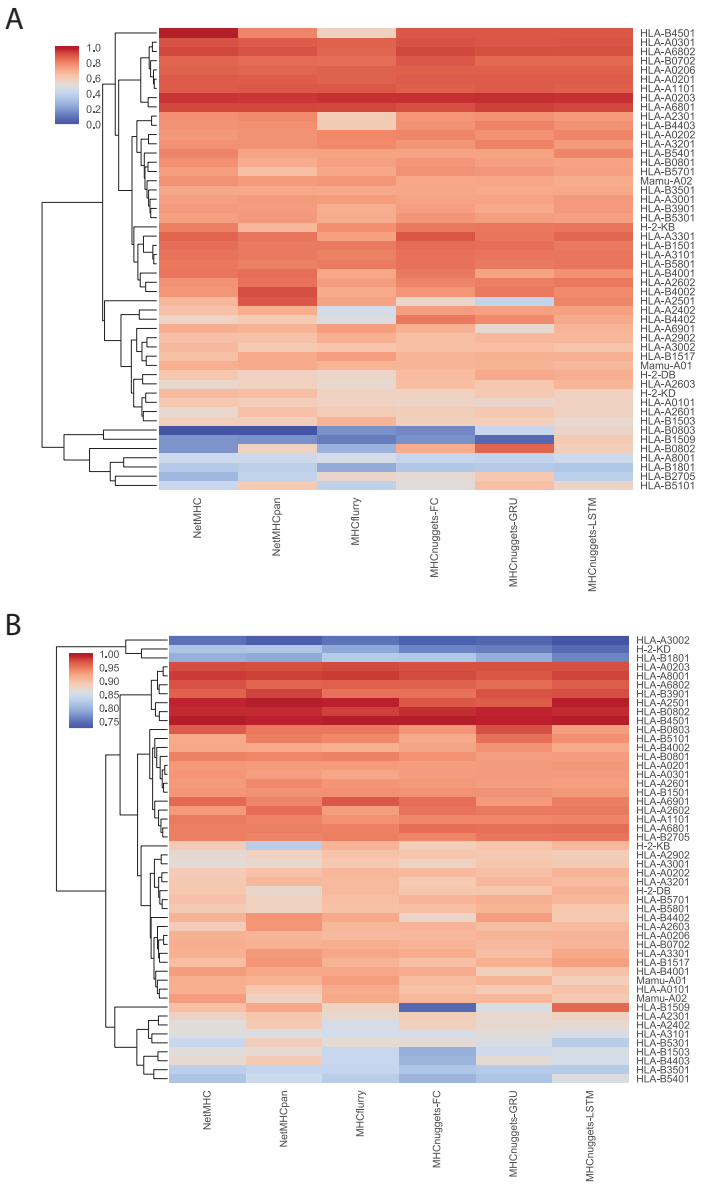

Figure S1: ***In silico* prediction performance of the six best methods varies considerably by individual MHC Class I allele but does not vary much by prediction method.** A. F1 scores B. AUC. In the hierarchically clustered heat map, the methods index the columns and alleles index the rows. Heat map coloring indicates high (red) vs. low (blue) F1 scores or AUC. The high variability by allele is seen by looking up or down any column. The low variability by method is seen by looking across any row.

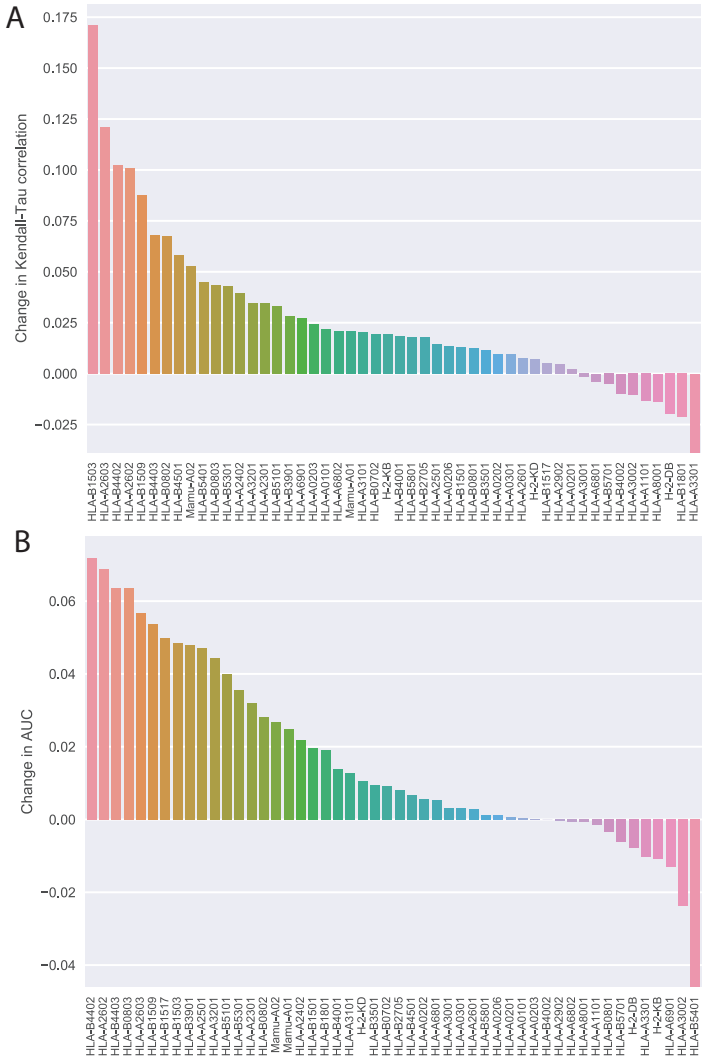

Figure S2: **Transfer learning improves overall performance for neural networks.** **A**. Kendall-Tau correlation. **B**. AUC. Comparison is between deep learning MHCnuggets-GRU networks trained with and without the transfer learning protocol. The 50 MHC class I alleles in the Kim benchmark set are shown and the bar height and color represent the net change Kendall-Tau correlation or AUC after transfer learning was applied.

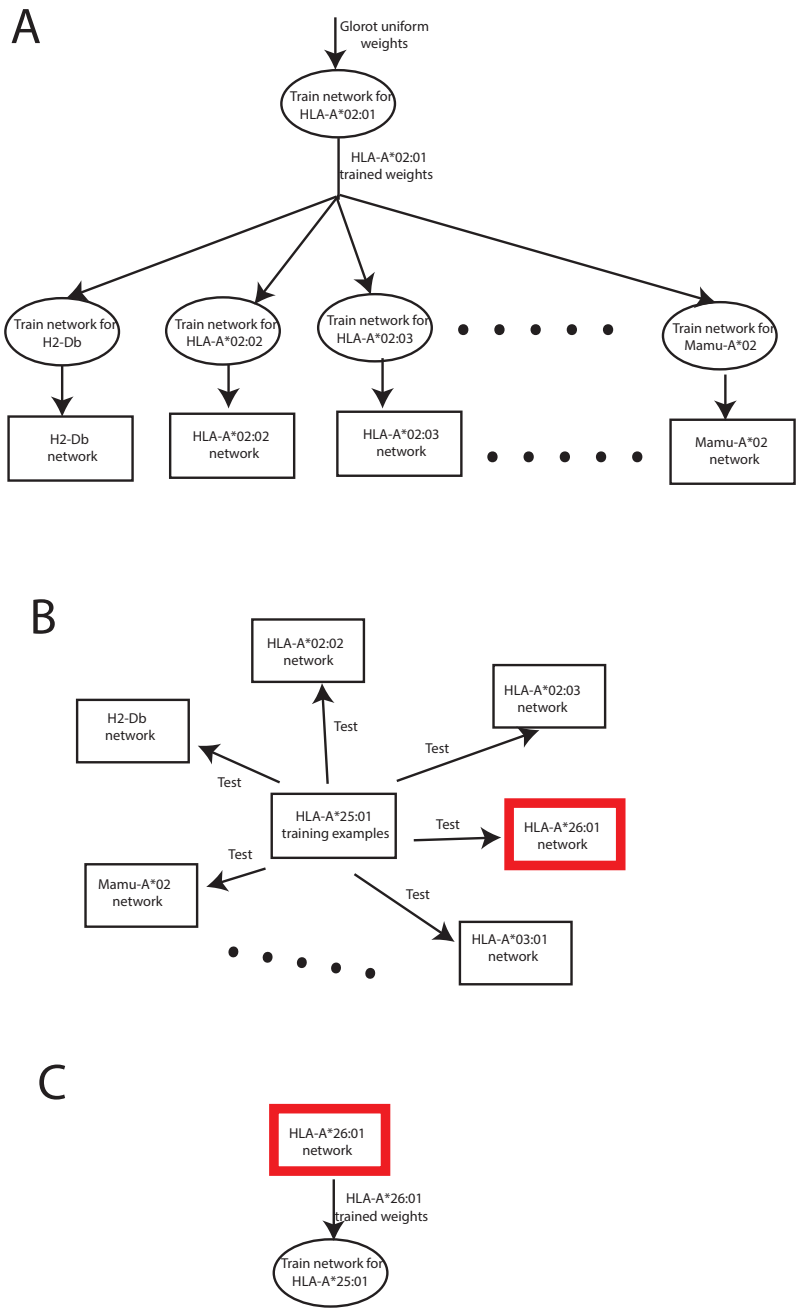

Figure S3: **Schematic of transfer learning protocol. A.** Network weights of HLA-A*02:01 are used to initialize 50 networks for all other alleles in the Kim benchmark test set. **B.** The training examples for each allele are tested for performance using all the networks trained in A. Example for a single allele HLA-A*25:01 is shown. The network with the highest AUC (originally trained on HLA-A*26:01) is selected (red highlight). **C.** The network weights for the best networks found in the previous step (e.g., HLA-A*26:01) are used to initialize and retrain the network for each allele of interest.

Supplementary Materials and Methods

MHCnuggets technical details

Peptide representation

Each residue is represented as a 21-dimensional smoothed, one-hot encoded vector (0.9 and 0.005 replace 1 and 0).

Optimization

All MHCnuggets networks were trained with backpropagation using the Adam optimizer [46], with a learning rate of 0.001. The activation function for all intermediate network layers was *tanh*. The final output layer of all architectures was a single sigmoid unit, related to the predicted binding affinity as $y = max(0, 1 - log_{50K}\text{IC50})$. For all networks, the number of hidden units, dropout [47] rate, and number of training epochs was estimated by three-fold cross-validation on the HLA allele with the largest number of entries (HLA-A*02:01).

Artificial peptide shortening and lengthening made efficient

Convolutional neural networks and standard fully-connected neural networks require artificial shortening and lengthening of inputs. For these methods, any peptides that were longer or shorter than 9 residues were cut or padded. To minimize the noise added by systematic padding and cutting operations, these operations were applied only once to each peptide and only to positions 6 and/or 7, which minimized the impact on the critical primary and secondary anchor residue positions.

Convolutional neural networks

MHCnuggets-Chunky-CNN is a 1D convolutional neural network [27], with kernel sizes (2,3), stride=1, number of filters =(250,250), a global max pooling layer, a fully connected layer with 64 hidden units, and a dropout probability of 0.6, trained for 100 epochs. MHCnuggets-Spanny-CNN is a 1D convolutional neural network, with kernel sizes (2,3,9), stride=1, number of filters =(1,1,1), a global max pooling layer, a fully connected layer with 64 hidden units, and a dropout probability of 0.6, trained for 100 epochs.

Implementation

All networks were trained using the Keras python package (TensorFlow back-end) [48] [49].