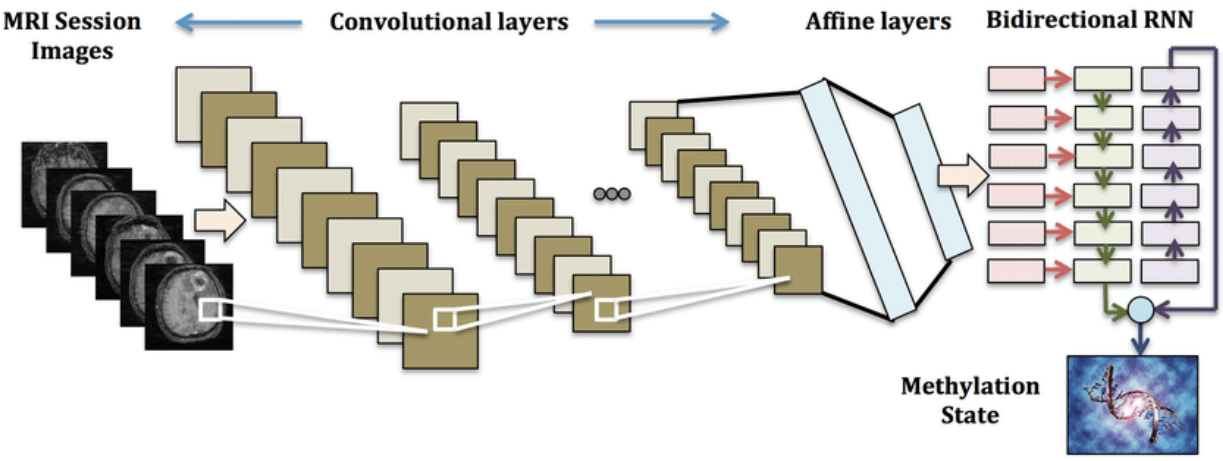


Your paper:

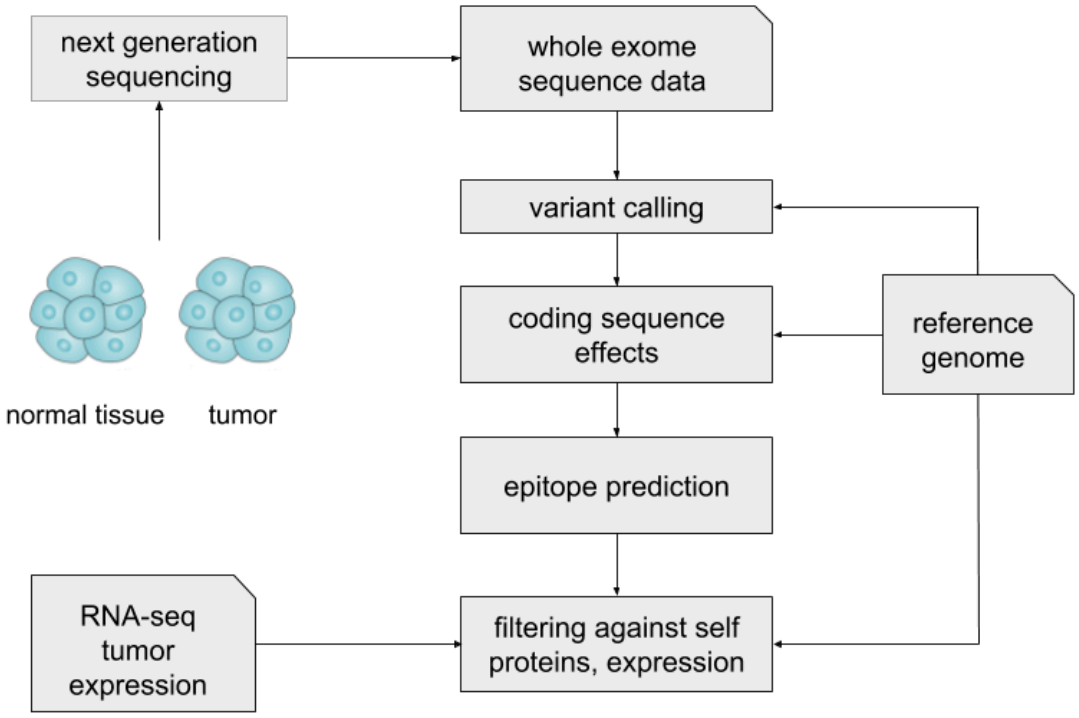
MHCherryPan, a novel model to predict the bindingaffinity of pan-specific class I HLA-peptide

(you can check your upload file by clicking on your paper title here)

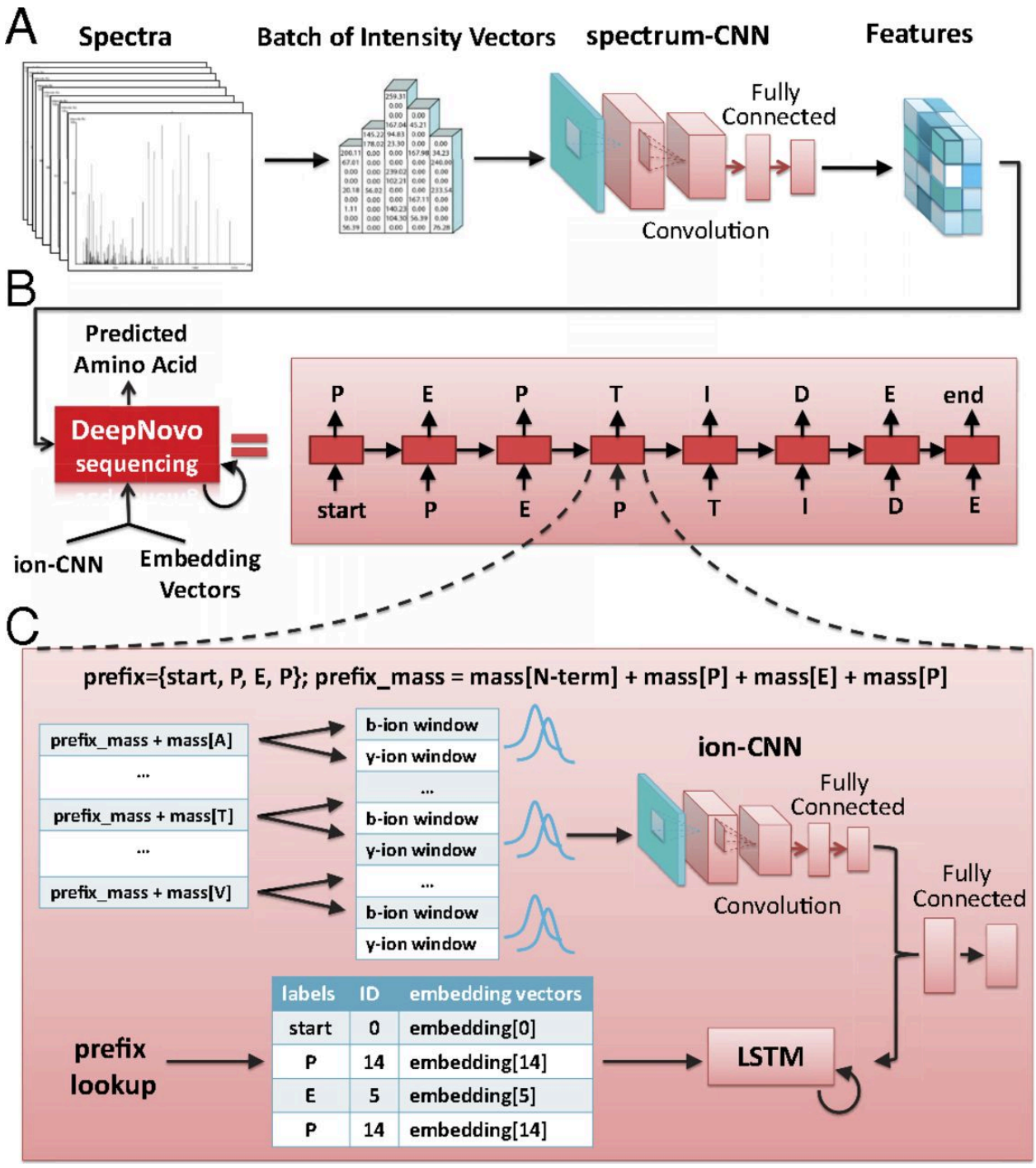
has been received. Please remember the paper ID below:

Paper ID: B522

Paper Password: p5222616



https://epitopepredict.readthedocs.io/en/latest/examples.html


A
Spectra
Batch of Intensity Vectors
spectrum-CNN
Features
Fully Connected
Convolution
B
Predicted Amino Acid
DeepNovo sequencing
ion-CNN
Embedding Vectors
P E P T I D E end
start P E P T I D E
C
prefix={start, P, E, P}; prefix_mass = mass[N-term] + mass[P] + mass[E] + mass[P]
prefix_mass + mass[A]
...
prefix_mass + mass[T]
...
prefix_mass + mass[V]
b-ion window
y-ion window
...
b-ion window
y-ion window
...
b-ion window
y-ion window
ion-CNN
Fully Connected
Convolution
Fully Connected
prefix lookup
labels ID embedding vectors
start 0 embedding[0]
P 14 embedding[14]
E 5 embedding[5]
P 14 embedding[14]
LSTM

My Qs:

IC50:

Binary:

Benchmark Evaluations:

Sharcnet:

UserName: kzhang@uwo.ca

Password: Wocs1989

What does masking do?

Masking allows us to handle variable length inputs in RNNs. Although RNNs can handle variable length inputs, they still need fixed length inputs. Therefore, what we do is to create a mask per sample initialised with 0 with a length equal to the longest sequence in the dataset. Then we fill the mask with 1s to all the position where the sample has values in.

For example,

The longest sequence in a dataset is 10, thus the masks will be initialised as follows:

mask = [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]

we also have a sample:

a = [2., 0. ,5. ,6.]

now we fill the mask with ones to all the positions a has values in. Therefore we get the following mask:

mask_a = [1., 1., 1., 1., 0., 0., 0., 0., 0., 0.]

After that, we feed the sample and the mask to a RNN. Under the hood, the RNNs is going to add 0 to all the positions a doesn't have values in so *a* will turn into

a_hood = [2., 0. ,5. ,6., 0., 0., 0., 0., 0., 0.]

However, if we leave it as it is, the RNN is going to think the sequence has length 10 and use all the appended 0s.

mask_a = [1., 1., 1., 1., 0., 0., 0., 0., 0., 0.]

The *mask_a* is going to be used to skip any input with mask 0 by copying the previous hidden state of the cell; it will proceed normally for any input with mask 1.

I hope the answer helps.

9.1k views · View Upvoters

https://www.quora.com/What-is-masking-in-a-recurrent-neural-network-RNN

Updated on June 22nd

Prof Zhang's idea. What happened if we remove all the peptides that are normal only targeting on those potential tumor cells? If we only use that to train our model, will the result be better or worse?

Updated: After about 2 days. AUC is still around 0.88 for classification and regression is increased from 0.5 to 0.69 after one day training.

Meeting on June 21

1. Padding. Confirm with prof. And the value (* -4 -4 -4 -4 -4) for padding.
2. CNNLSTM
3. Statistics

What does LSTM return?

you're confusing cells with units. He describes a single cell. Units refers to the dimensionality of your hidden state and thus the dimensionality of your weight matrices (roughly)

Think of a single normal fully connected layer. Your input matrix has the dimensions (n_samples x n_features) and your weight matrix is then (n_features x n_units). So you if you have a sample with 2048 features and pass it through a fully connected layer with 512 "units", then your output of that layer will now have 512 features.

Same general idea for your weight matrices in an LSTM.

In more detail, there are 2 matrices for 3 gates + cell state (one with dimensions n_input_features x n_units and one with n_units x n_units).

so 8 matrices altogether in a cell. # of weights (not including bias term) = 4(n_input_features x n_units + n_units^2)

Now let's talk about how the pipeline works at a high level:

Into the LSTM cell, we feed in timestep 1 by itself. The input is a vector with size n_input_features. This is fed through all our fancy gates and stuff and we are left with a hidden state and cell state

Then we go into the next LSTM cell (which is actually just the same "cell", we're just looping back to the entrance of it). This time though our input will be the last hidden state, cell state, and now timestep 2.

This repeats for each time step. So now we have a hidden state vector for each time step.

We started with timesteps x n_input_features and finish with timesteps x n_units.

But commonly people will just take the hidden state vector of the final timestep as the output of their LSTM instead of keeping all the hidden states. So that's how we go from timesteps x n_input_features all the way to a single vector with size n_units

https://www.reddit.com/r/MachineLearning/comments/87djn7/d_what_is_meant_by_number_of_hidden_units_in_an/

When the return_sequences argument is set to False (default), the network will only output hn, i.e. the hidden state at the final time step. Otherwise, the network will output the full sequence of

hidden states, [h1, h2, ..., hn]. The internal equations of the layer are unchanged. Refer to the documentation.

Understand the Difference Between Return Sequences and Return States for LSTMs in Keras

https://machinelearningmastery.com/return-sequences-and-return-states-for-lstms-in-keras/

In `keras` - while building a sequential model - usually the second dimension (one after sample dimension) - is related to a `time` dimension. This means that if for example, your data is `5-dim`with `(sample, time, width, length, channel)` you could apply a convolutional layer using `TimeDistributed` (which is applicable to `4-dim` with `(sample, width, length, channel)`) along a time dimension (applying the same layer to each time slice) in order to obtain `5-d` output.

The case with `Dense` is that in `keras` from version 2.0 `Dense` is by default applied to only last dimension (e.g. if you apply `Dense(10)` to input with shape `(n, m, o, p)` you'll get output with shape `(n, m, o, 10)`) so in your case `Dense` and `TimeDistributed(Dense)` are equivalent.

`Padding`
`https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction/`

`numpy.pad` with `constant` mode does what you need, where we can pass a tuple as second argument to tell how many zeros to pad on each size, a `(2, 3)` for instance will pad **2** zeros on the left side and **3** zeros on the right side:

Given `A` as:

https://stackoverflow.com/questions/38191855/zero-pad-numpy-array/38192105

1

As Keras documentation suggests TimeDistributed is a wrapper that applies a layer to every temporal slice of an input.

Here is an example which might help:

Let's say that you have video samples of cats and your task is a simple video classification problem, returning 0 if the cat is not moving or 1 if the cat is moving. Let's assume your input dim is (None, 50, 25, 25, 3) which means you have 50 time steps or frames per sample, and your frames are 25 by 25 and have 3 channels, rgb.

Well, one aporoach would be to extract some "features" from each frame using CNN, like Conv2D, and then pass them to an LSTM layer. But the feature extraction would be the same for each frame. Now TimeDistributed comes to the rescue. You can wrap your Conv2D with it, then pass the output to a Flatten layer wrapped also by TimeDistributed. So after applying TimeDistributed(Conv2D(...)), the output would be something of dim like (None, 50, 5, 5, 16), and after TimeDistributed(Flatten()), the output would be of dim (None, 50, 400). (The actual dim would depend on Conv2D parameters.)

The output at this layer now can be passes through LSTM.

So obviously, LSTM itself does not need a TimeDistributed wrapper.

Now CNN+LSTM model

https://towardsdatascience.com/get-started-with-using-cnn-lstm-for-forecasting-6f0f4dde5826

https://medium.com/@shivajbd/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e

https://github.com/vanessajurtz/lasagne4bio/blob/master/subcellular_localization/notebook%20tutorial/CNN-LSTM-Attention.ipynb

TimeDistributedDense对3D张量的每个时间步应用相同的Dense（完全连接）操作。

https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/

Meeting June 7th

How padding?

Analysis on 9-aa and 10-aas. The most important positions seem on 2nd and the rightest.

Updated on June 7th

APBC 2020 : The 18th Asia Pacific Bioinformatics Conference.

https://blog.csdn.net/mitedu/article/details/8480111

https://blog.csdn.net/lydia2012924/article/details/77983749

APBC 2020 : The 18th Asia Pacific Bioinformatics Conference

http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=86176&copyownerid=89249

ICMLA 2019 : 18th IEEE International Conference on Machine Learning and Applications

http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=83617

Go to Waterloo company on Fri. May 31st

Qs:

1. Those parameters. Time-in-steps. Features. TimeDistributed. LSTM
2. How to optimize the parameters.
3. ROC-AUC calculation.
4. Combination with CNN
5. Attention model.
6. Blosum 62.
7.

The encoder-decoder recurrent neural network is an architecture where one set of LSTMs learn to encode input sequences into a fixed-length internal representation, and second set of LSTMs read the internal representation and decode it into an output sequence.

This architecture has shown state-of-the-art results on difficult sequence prediction problems like text translation and quickly became the dominant approach.

MHC iedb why positive, positive high, positive low ????????

The HLA sequence was encoded in terms of a pseudo-sequence consisting of amino acid residues in contact with the peptide. The contact residues are defined as being within 4.0 Å of the peptide in any of a representative set of HLA-A and -B structures with nonamer peptides. Only polymorphic residues from A, B, and C alleles were included giving rise to a pseudo-sequence consisting of 34 amino acid residues. Notice that due to multiple possible conformations, the central peptide residues could choose to interact with different subsets of residues in the binding groove. All such residues were included in the pseudo-sequence. The interaction map between the peptide and HLA sequence is given in Figure 4.

https://www.ncbi.nlm.nih.gov/pubmed/17726526/

Dataset

The MHC class I molecule was represented by a pseudo-sequence consisting of amino acid residues in contact with the peptide. The contact residues are defined as being within 4.0 Å of the peptide in any of a representative set of HLA-A and HLA-B structures binding a
nonameric peptide. Of all contact residues, only those that were polymorphic in any known HLA-A, HLA-B, and HLA-C protein sequence were included, giving rise to a pseudosequence consisting of 34 amino acid residues (Nielsen et al. 2007). This pseudo-sequence
mapping was applied to all MHC molecules in this study. This could lead us to discard essential peptide–MHC interactions for non-classical and non-human MHC molecules.
However, no quantitative peptide-binding data are available for non-classical HLA molecules, and only very limited data are available for non-human primates. The panspecific approach relies on the ability of the neural networks to capture general features of the relationship between peptides and HLA pseudo-sequences and interpret these in terms of binding affinity. Only interactions that are polymorphic in the training data

can aid the neural network learning. It would hence not be possible for the NetMHCpan method to learn from such extended pseudo-sequence mappings due to the lack of polymorphism at the extended MHC positions in the training data

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3319061/pdf/nihms-138465.pdf

Assay Information

Assay

The **Assay Finder** is used to select the **Assay Type**, **Assay Type Group**, and **Assay Type Units**. The units available are EC50 nM, IC50 nM, t1/2 (min), Other (see comments)/None, KD nM, Koff (s^-1), Tm (degree C), and Angstroms. Be sure to utilize the correct units for the assay you are entering.

Qualitative Measurement The Qualitative Measurement field is a ***required field*** . Select the appropriate response based upon the figures and text provided in the manuscript. All assay contexts in the database must be labeled as either positive or negative. At times, this can be controversial. The overriding rule is to follow what the author states, however, data may be present without comment from the authors regarding the result. For truly confusing data, the author may be contacted for clarification. For data that can be reasonably considered negative or positive based upon other information provided in the manuscript, the curator may make a judgment. In most cases, the presence of a precise numerical value for MHC binding contexts is curated as a positive outcome.

It consist of a drop down menu of:

- Positive
- Positive-Low
- Positive-Intermediate
- Positive-High
- Negative

Rule 1: When authors specify a qualitative assessment in the reference as either positive or negative, their assessment will be recorded in the database.

Rule 2: When a qualitative assessment can be inferred from the information in the manuscript (threshold is provided), this assessment will be entered into the database.

Rule 3: When no qualitative assessment is provided by the authors, this data will not be entered into the database.

Measurement Inequality These selections may be used in order to add additional information regarding the quantitative value.

Quantitative Measurement Whenever quantitative data are available, they must be captured. Always capture quantitative values, when provided, for MHC binding assays. Be sure to select the appropriate units and check your math if conversions are required.

Always use author statements to determine which values are considered positive or negative. Different assays and different authors use different criteria for positive or negative binding values.

- MHC Binding data for select authors:Alex, John Sidney, Buus, Lund.

Class I

IC50 < = 50nM - Positive-High

51 to <= 500nM- Positive -Intermediate

501 <= 5000nM - Positive-Low

>5000nM - Negative

Class II

IC50 < = 100nM - Positive-High

101 to <= 1000nM - Positive -Intermediate

1001 <= 10000nM - Positive-Low

>10000nM - Negative

http://curationwiki.iedb.org/wiki/index.php/Curation_Manual2.0

MHC Binding Assay Fields

Experimental data characterizing the interaction between an epitope and an MHC molecule is entered under the tab labeled MHC binding.

MHC Fields

The fields utilized to capture MHC binding contexts are as follows:

Location of Data This field is used to identify where the MHC binding data appears. All tables, figures, or text where the actual binding data is provided should be entered. If bulking different figures or tables, enter all of them here.

Assay Information

Assay

The **Assay Finder** is used to select the **Assay Type**, **Assay Type Group**, and **Assay Type Units**. The units available are EC50 nM, IC50 nM, t1/2 (min), Other (see comments)/None, KD nM, Koff (s^-1), Tm (degree C), and Angstroms. Be sure to utilize the correct units for the assay you are entering.

Qualitative Measurement The Qualitative Measurement field is a ***required field*** . Select the appropriate response based upon the figures and text provided in the manuscript. All assay contexts in the database must be labeled as either positive or negative.

It consist of a drop down menu of:

- Positive
- Positive-Low
- Positive-Intermediate
- Positive-High
- Negative

Measurement Inequality These selections may be used in order to add additional information regarding the quantitative value.

Quantitative Measurement Whenever quantitative data are available, they must be captured. Always capture quantitative values, when provided, for MHC binding assays. Be sure to select the appropriate units and check your math if conversions are required.

Assay Comments This field is used to enter comments relevant to the binding assay such as comparisons made between the binding of an analog or mutant epitope to the binding of the native epitope. As with all comments fields, this field should only be used when necessary. The comments should be a complete “stand alone” concept written in proper English and must provide information of value that would otherwise be lost given the current IEDB fields.

MHC Allele Name

The Allele name is selected from the Allele Finder application. This table includes Allele Name, Class, Organism, Restriction level, Haplotype, Locus, Serotype, and Molecule. Each of the table headings can be used to order the table. The search window allows one to search by organism, class, and allele. Synonyms will also be found so if you enter an allele name into the finder and a different allele name is returned, be sure to note that they are synonyms.

http://curationwiki.iedb.org/wiki/index.php/Curation_Manual2.0

The assay types present within the IEDB reflect the assays being used in the literature. This list includes all commonly used immunological methods such as ELISA, FACs, bioassays, etc. and expands to accommodate new assays as they are developed. For all assay types, the IEDB uses

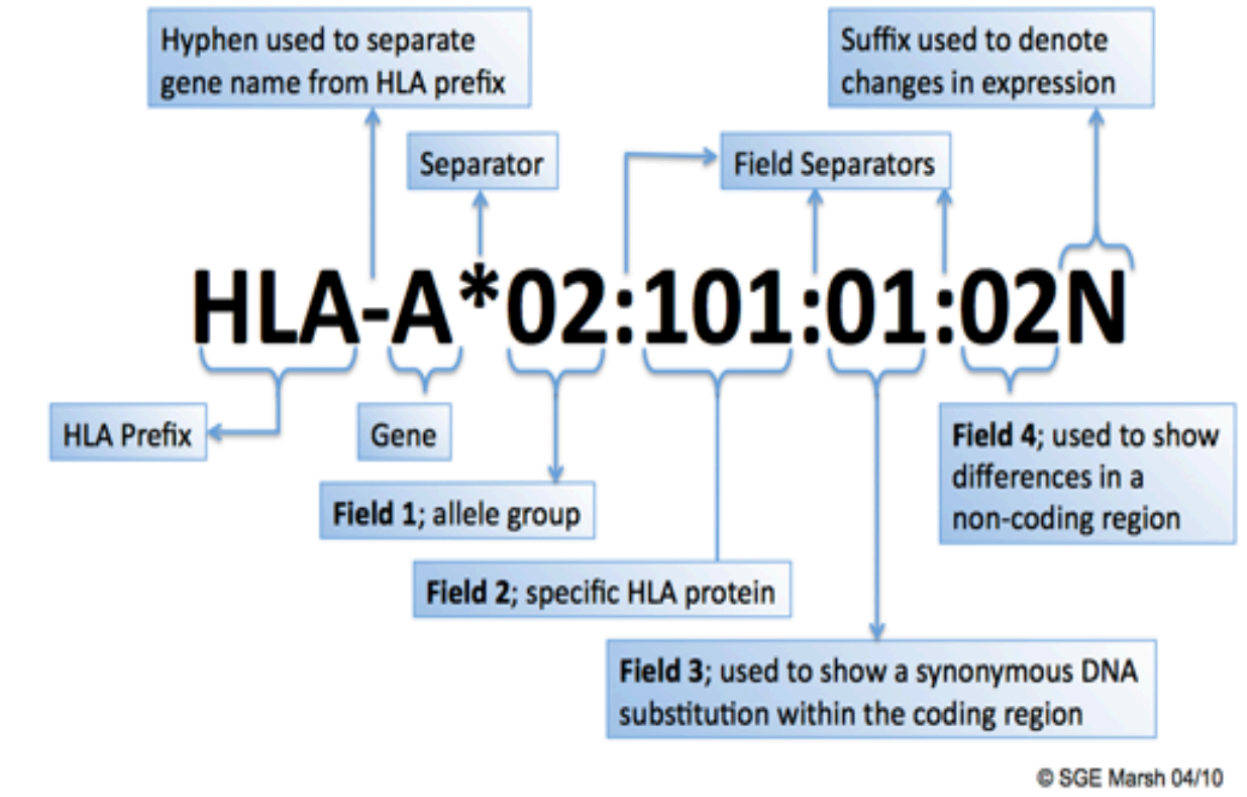
an assay finder application that allows users to search on either the purpose of the assay (to measure IL-2) or the method used (ELISA). All experimental data entered into the IEDB are categorized as either positive or negative in the 'Qualitative Measurement' field. Additional granularity is available for positive data with values of positive-high, positive-intermediate and positive-low. For assay types with quantitative measurements, such as the K_D between an antibody and epitope, the numerical values are entered along with the units. When available, the number of subjects tested, the number responding or the percent responding are also displayed.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808938/

Physicochemical encoding (PE): this encoding is particularly suited for peptides since it exploits the fixed length of the sequence. In (Zhao et al. (2003) each amino acid of the peptide is encoded by F = 10 factors concatenated (i.e. each amino acid is described by a vector a 2 RF , therefore the feature vector is composed by F M features. These orthogonal factors (sa) ai, i = 1, ... , 20 were obtained from 188 physical properties of 20 amino acids via multivariate statistical analyses (Kidera, Konishi, Oka, Ooi, & Scheraga, 1985). Several other encodings have been proposed based on a similar representation, but starting from a different set of factors (derived from the physicochemical properties). In Maetschke, Towsey, and Bodén (2005) the set of factors was obtained by reducing the BLOSUM62 matrix by Sammon projection (Henikoff & Henikoff, 1992) to a 20 F (F = 5) matrix (bs). In Venkatarajan and Braun (2001) a multidimensional scaling of 237 physicochemical properties is performed to derive F = 5 descriptors for all 20 amino acids (ms). In a recent work (Tong, Liu, Zhou, Wu, & Li, 2008) a novel descriptor of F = 9 principal component scores is derived from the principal component analysis of a matrix of 99 weighted holistic invariant molecular indices of amino acids (sw).

HLA Naming rule:
http://hla.alleles.org/nomenclature/naming.html

The digits before the first colon describe the type, which often corresponds to the serological antigen carried by an allotype. The next set of digits are used to list the subtypes, numbers being assigned in the order in which DNA sequences have been determined. Alleles whose numbers differ in the two sets of digits must differ in one or more nucleotide substitutions that change the amino acid sequence of the encoded protein. Alleles that differ only by synonymous nucleotide substitutions (also called silent or non-coding substitutions) within the coding sequence are distinguished by the use of the third set of digits. Alleles that only differ by sequence polymorphisms in the introns, or in the 5' or 3' untranslated regions that flank the exons and introns, are distinguished by the use of the fourth set of digits.



Meeting with the company, on May 24th:

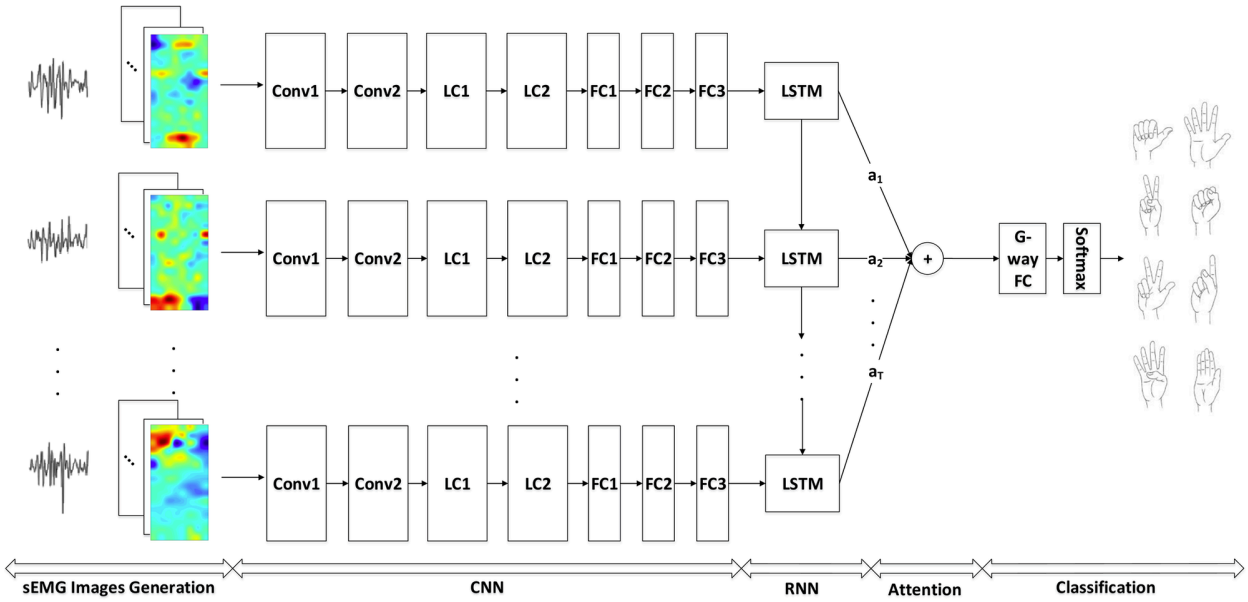
Qs for Baozhen:

1. 确认下自己对MHC的理解。

目前识别候选新抗原的普遍方法通常包括两个主要阶段：(i)癌症和正常组织的外显子组测序以发现体细胞突变和(ii)预测哪些突变肽最可能由MHC蛋白呈递用于T细胞识别

2. 确认一下他们input的数据，确认到是那篇文章

3. 问一下他们CNN 加LSTM具体的合在一起用的是什么层，以及他们一般怎么进行调参的



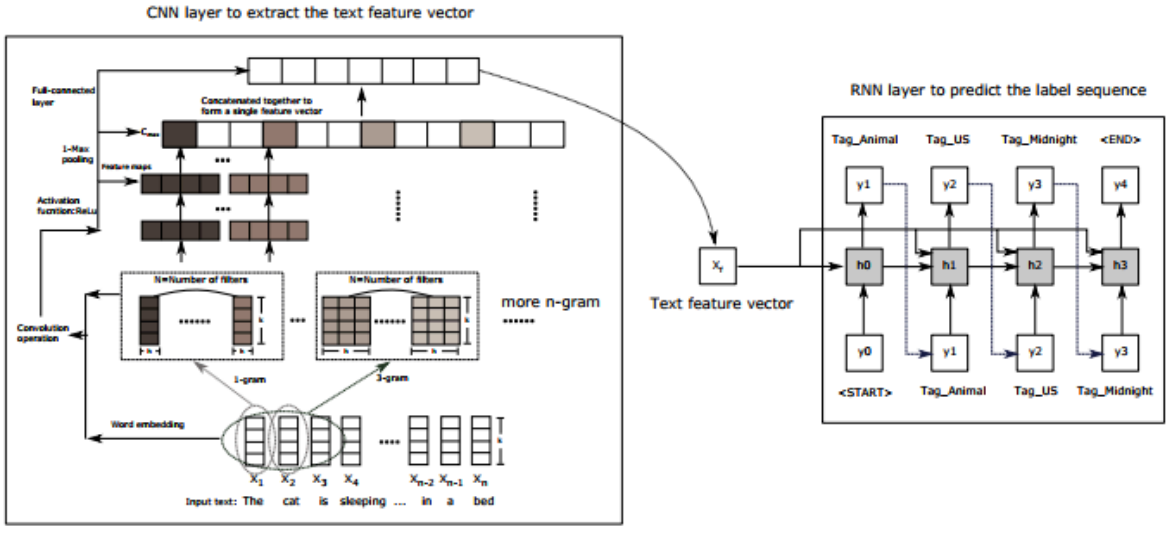
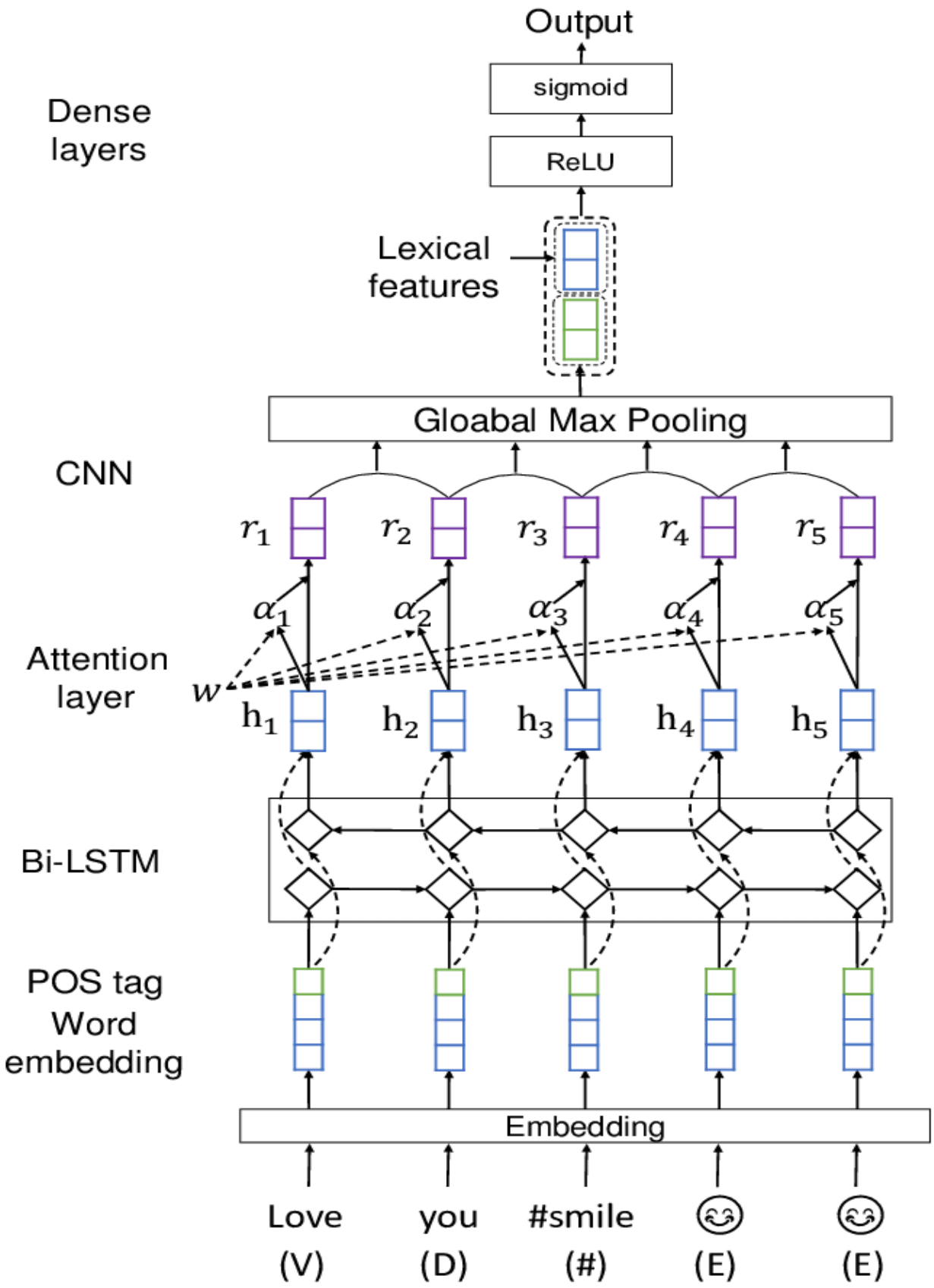
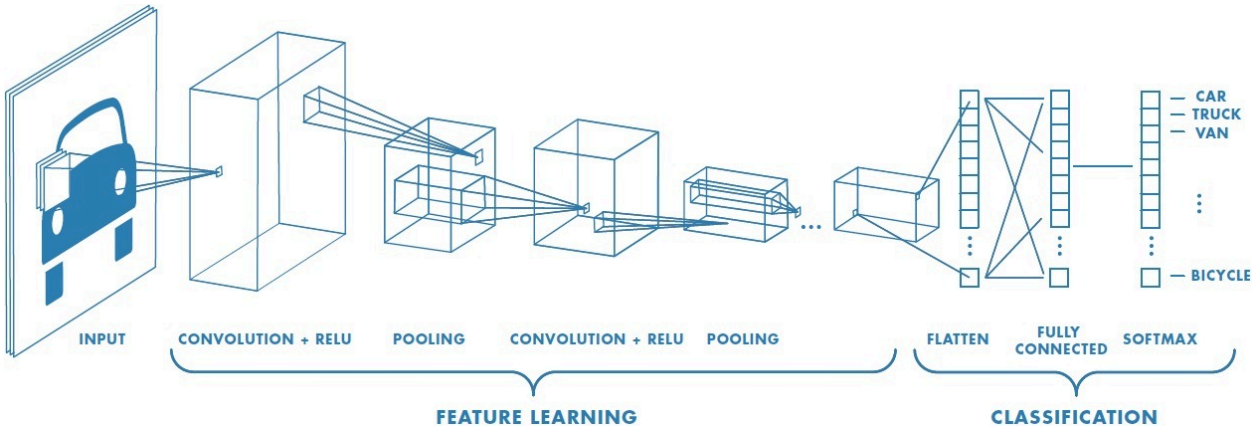
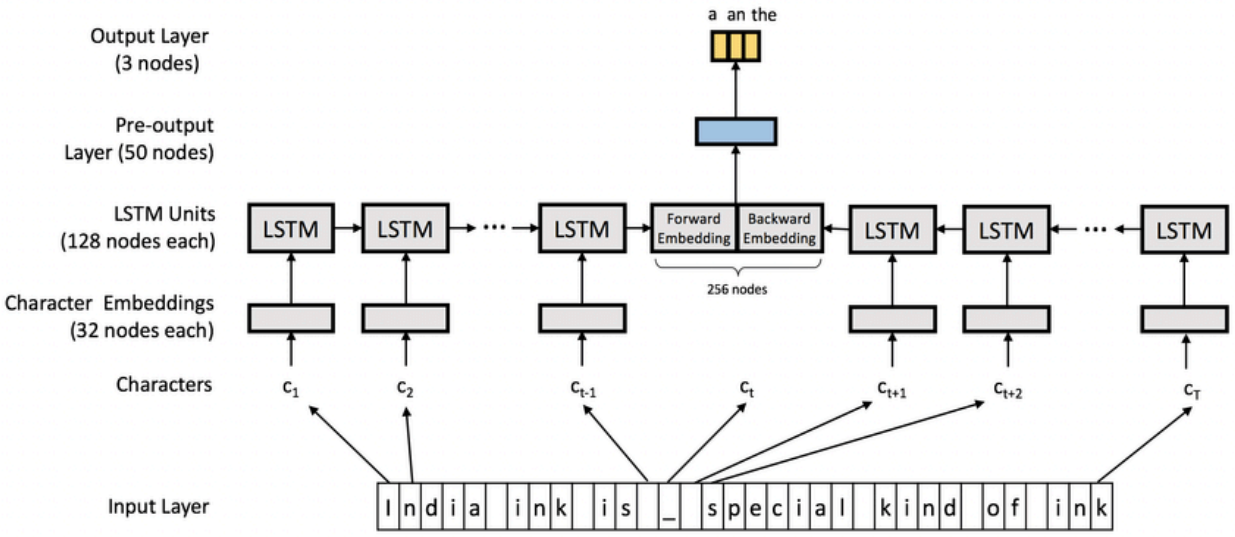
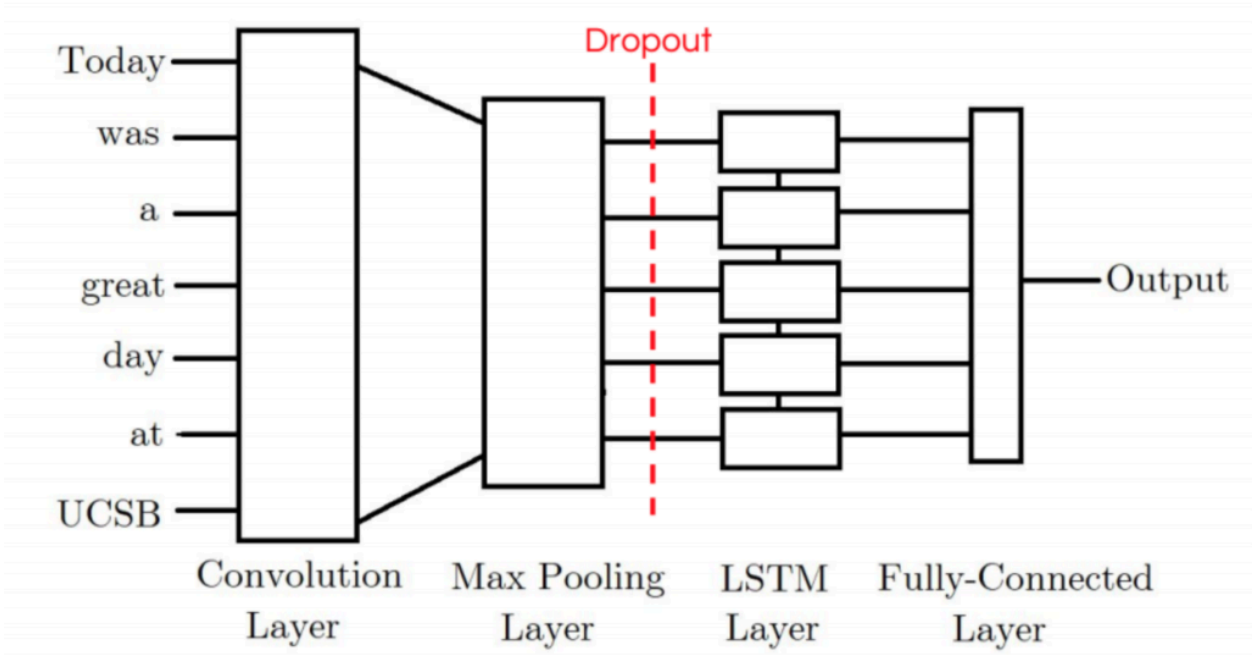


Fig. 1: Our proposed CNN-RNN model


Output
sigmoid
ReLU
Dense layers
Lexical features
Gloabal Max Pooling
CNN
r_1 r_2 r_3 r_4 r_5
α_1 α_2 α_3 α_4 α_5
Attention layer
w
h_1 h_2 h_3 h_4 h_5
Bi-LSTM
POS tag
Word embedding
Embedding
Love (V)
you (D)
#smile (#)
(E)
(E)


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


Output Layer (3 nodes)
Pre-output Layer (50 nodes)
LSTM Units (128 nodes each)
Character Embeddings (32 nodes each)
Characters
Input Layer
a an the
LSTM
Forward Embedding
Backward Embedding
256 nodes
c_1 c_2 c_{t-1} c_t c_{t+1} c_{t+2} c_T
I n d i a i n k i s _ s p e c i a l k i n d o f i n k



RNN

[samples, time_steps, features]

Samples - This is the len(dataX), or the amount of data points you have.

Time steps - This is equivalent to the amount of time steps you run your recurrent neural network. If you want your network to have memory of 60 characters, this number should be 60.

Features - this is the amount of features in every time step. If you are processing pictures, this is the amount of pixels. In this case you seem to have 1 feature per time step.

2

I also had this question before. On a higher level, in `(samples, time steps, features)`

1. `samples` are the number of data, or say how many rows are there in your data set
2. `time step` is the number of times to feed in the model or `LSTM`
3. `features` is the number of columns of each sample

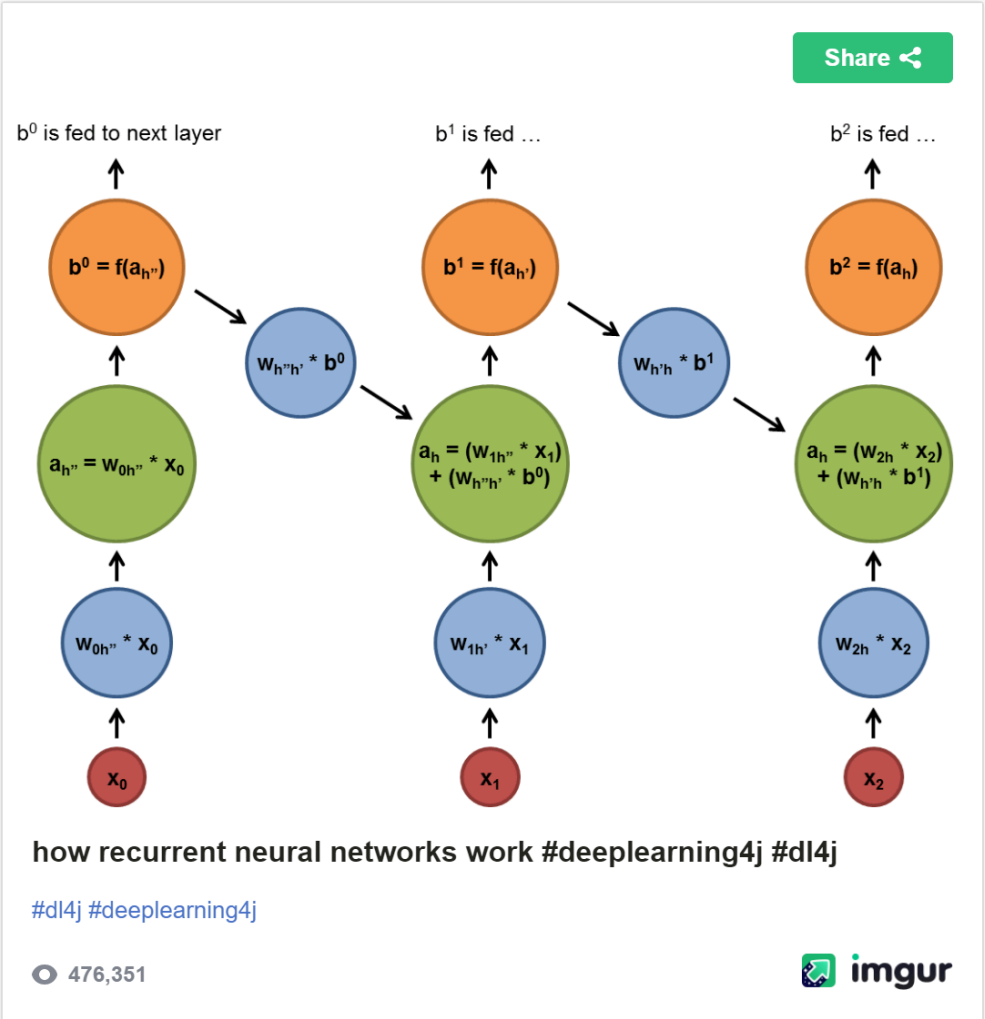
For me, I think a better example to understand it is that in `NLP`, suppose you have a sentence to process, then here sample is 1, which means 1 sentence to read, `time step` is the number of words in that sentence, you feed in the sentence word by word before the model read all the words and get a whole context of that sentence, `features` here is the dimension of each word, because in word embedding like `word2vec` or `glove`, each word is interpreted by a vector with multiple dimensions.

The `input_shape` parameter in `Keras` is only `(time_steps, num_features)`, more you can refer to this.

And the problem of yours is that when you reshape data, the multiplication of each dimension should equal to the multiplication of dimensions of original data set, where 434 does not equal to 217*3*2.

When you implement `LSTM`, you should be very clear of what are the features and what are the element you want the model to read each time step. There is a very similar case here surely can help you. For example, if you are trying to predict the value of time `t` using `t-1` and `t-2`, you can either choose to feed in two values as one element to predict `t`, where `(time_step, num_features)=(1, 2)`, or you can feed each value in 2 time steps, where `(time_step, num_features)=(2, 1)`.

That's basically how I understand this, hope make it clear for you.



Recurrent networks are distinguished from feedforward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory.2 Adding memory to neural networks has a purpose: There is information in the sequence itself, and recurrent nets use it to perform tasks that feedforward networks can't.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before. One way to think about RNNs is this: they are a way to share weights over time.

Updates on May 20th.

Meeting Prof Zhang on May 20th.
BGI recently published a paper using Denovo to predict binding affinity of MHC. It is pre-printed on: https://www.biorxiv.org/content/biorxiv/early/2019/04/26/620468.full.pdf . Briefly summary, it used MS data directly trained on Deepnovo to predict binding of MHC.

由于新抗原携带肿瘤特异性突变而在正常组织中未发现，它们是免疫系统区分癌细胞和非癌细胞的理想靶标。

目前识别候选新抗原的普遍方法通常包括两个主要阶段：(i)癌症和正常组织的外显子组测序以发现体细胞突变和(ii)预测哪些突变肽最可能由MHC蛋白呈递用于T细胞识别。

前的计算机方法着重于预测哪些肽在给定患者的HLA等位基因时结合MHC蛋白，例如，NetMHC。然而，只有极少数来自大量预测的候选物被证实存在于肿瘤细胞表面，甚至更少被发现引发T细胞应答。已经进行了若干努力来改进MHC结合预测，包括使用质谱数据以及结合亲和力数据以更准确地预测MHC抗原呈递。最近，已经提出蛋白质组学方法将质谱法和外显子组测序组合以鉴定直接从MHC蛋白分离的新抗原，从而克服MHC结合预测的局限性。在那些方法中，外显子组测序是进行构建定制的蛋白质数据库，包括所有正常和突变的蛋白质序列。搜索引擎进一步使用该数据库来鉴定通过免疫沉淀测定和质谱法获得的内源肽，包括新抗原。然而，现有的数据库搜索引擎不是针对HLA肽而设计的，并且偏向于胰蛋白酶肽。此外，当处理由(i)来自外显子组测序的所有mRNA同种型和(ii)HLA肽的未知消化规则所产生的非常大的搜索空间时，它们具有敏感性和特异性问题。外显子组测序错误和偏差也可以累积到数据库搜索过程中。

https://www.nature.com/articles/nbt.3800 The problem with neoantigen prediction. 简介MHC和 neoantigen.

新表位发现的关键问题是突变蛋白质被蛋白酶体加工成8至11个残基的肽，通过与抗原加工(TAP)相关的转运蛋白穿梭到内质网中并加载到新合成的主要组织相容性复合物I类(MHC-I)用于CD8 + T细胞的识别。

尽管一些计算方法集中于预测在抗原加工(例如，NetChop)和肽转运(例如，NetCTL)期间发生的情况，但是大多数努力集中于模拟哪些肽与MHC-I分子结合。基于神经网络的方法，例如NetMHC，用于预测产生适合患者MHC-I分子凹槽的表位的抗原序列。可以应用其他过滤器以使假设蛋白质优先化并且测量突变的氨基酸是否可能定向为面向MHC(朝向T细胞受体)或降低表位对MHC-I分子本身的亲和力。

各种混淆因素意味着这些预测可能会出错。 测序已经在用作肽的起始材料的读段中引入了扩增偏差和技术错误。 表位处理和呈递的建模还必须考虑到人类具有约5,000个编码MHC-I分子的等位基因的事实，其中个体患者表达多达六个，均具有不同的表位亲和力。诸如NetMHC的方法通常需要50-100个实验确定的特定等位基因的肽结合测量，以建立具有足够准确度的模型。但由于

许多MHC等位基因缺乏此类数据，因此能够基于具有相似接触环境的MHC等位基因具有相似结合特异性来预测结合物的“泛特异性”方法越来越突出。

组会（May 17th, 2019）：

1. 选课
2. 汇报最近结果
3. 下周开会，提前复习文章，写一些问题问问。

问题：

预测MHC 数据库，如果我们需要找的是特殊病人的抗原，然后如果用IEDB 数据库进行training，数据是不是合不合预测，是不是应该用这个进行预测，以及我们的目标是不是找特殊出现的抗原这些抗原没有在数据库中出现但是在这些病人的sample 里出现了。

人体本身有的antigen，就是所有人都有的抗原并不是我们所需要的，我们主要是要找比如特殊疾病的病人sample 有的，但是是一般其他人没有的。那样我们用数据库来预测那些结合的还有意义吗 好像是叫 Neoantigens

抗原：（antigen，缩写Ag）是指能引起抗体生成的物质。它为任何可诱发免疫反应的物质。外来分子可经过B细胞上免疫球蛋白的辨识或经抗原呈现细胞的处理并与主要组织相容性复合体结合成复合物再活化T细胞，引发连续的免疫反应

表位: epitope，是存在于抗原表面的，决 定抗原特异性的特殊性结构的化学基团称为抗原决定族，又称表位。抗原通过表位与相应淋巴细胞表面抗原 受体结合，从而激活淋巴细胞，引起免疫应答；抗原也借此与相应抗体或致敏淋巴细胞发生特异性结合。个抗原分子可具有一种 或多种不同的表位，其大小相当于相应抗体 的抗原结合部位，每种表位只有一种抗原特 异性。因此，表位是被免疫细胞识别的靶结构，也是免疫反应具有特异性的基础，其性 质、数目和空间构型决定着抗原的特异性
抗体：

May 17th, 2019

Built up a CNN model with Keras by myself. Tested on https://github.com/ditekunov/mhc-peptides-dataset/tree/master/data dataset, achieved ACC 80.4%

May 15th, 2019

Thesis - Pan model to predict the binding affinity of MHC (or HLA).

Raw Data:

The input data is from several sources:
IEDB library: http://www.iedb.org/doc/mhc_ligand_full.zip
http://tools.iedb.org/mhci/download/ Kim et al (2014)

GitHub:
MHCflurry: https://github.com/openvax/mhcflurry/tree/master/downloads-generation
mhc-peptides-dataset https://github.com/ditekunov/mhc-peptides-dataset/tree/master/data

Deep-Learning-MHCI: https://github.com/altayg/Deep-Learning-MHCI/tree/master/dataset1

Epic: https://github.com/BGI2016/EPIC

Model (Code):
The code original is from the article - Ching T et al. 2018 Opportunities and obstacles for deep learning in biology and medicine. J.R.Soc. The github link:
https://github.com/vanessajurtz/lasagne4bio/tree/master/peptide_MHCII/scripts

This article generally summary and discuss the current hot areas where machine learning could be applied in biology. The article discussed a little bit about current popular software for MHC prediction (in page 18, under 3.9. Major histocompatibility complex-peptide binding). The article also provide a rough general model to predict the binding affinity for MHCII using Lasagne library. The code gives a general picture for how to use Lasagne to do this. The parameters (like hidden layers, filter sizes and so on) are not optimized and there are no performance evaluation for it (more like a general guide for how to use Lasagne and).

I currently modified the code to fit for my MHCI data.

CNN

Inputs: X: peptides, MHC (encoded as 34 AA pseudo sequences)
Y: binding or not binding. Or binding affinity

Model: Peptides, and MHC go through model. After each layer, eventually two inputs would become the layers with the same dimensions. After that, two inputs combines into one layer for final output.

Building the network...
input layer for peptide (l_in_pep.shape): (None, 21, None)
input layer for MHC pseudo sequence (l_in_mhc): (None, 21, 34)

DENSE ----------
elementwise sum instead of concat:

l_dense_conv_pep_9: (None, 60)
l_dense_mhc: (None, 60)
l_dense_all(combined): (None, 60)
# params: [W, b, W, b, W, b, beta, gamma, mean, inv_std, beta, gamma, mean, inv_std, W, b]
# weights: [0, 2, 4, 14]
# number of layers: 14
# number of parameters: 51001

X_pep_train.shape: (90849, 21, 15)
X_mhc_train.shape: (90849, 21, 34)
N_FEATURES: 21
MHC_SEQ_LEN: 34

LSTM:

References:

Ching T et al. 2018 Opportunities and obstacles for deep learning in biology and medicine. J. R. Soc. Interface 15: 20170387. http://dx.doi.org/10.1098/rsif.2017.0387

O'Donnell, Timothy J., et al. "MHCflurry: open-source class I MHC binding affinity prediction." *Cell systems* 7.1 (2018): 129-132.

Hu, Yan, et al. "ACME: Pan-specific peptide-MHC class I binding prediction through attention-based deep neural networks." *bioRxiv* (2018): 468363.

Liu, Zhonghao, et al. "DeepSeqPan, a novel deep convolutional neural network model for pan-specific class I HLA-peptide binding affinity prediction." *Scientific reports* 9.1 (2019): 794.

Kim, Yohan, et al. "Dataset size and composition impact the reliability of performance benchmarks for peptide-MHC binding predictions." *BMC bioinformatics* 15.1 (2014): 241.

Backup

The first stage in neoantigen identification from a VCF file is the proper annotation of variants to identify non-synonymous variants. To this end, NeoPredPipe employs the widely used and efficient genomics tool, ANNOVAR ([12]). Specifically, ANNOVAR processes samples in a way that prioritizes exonic variants, this step provides a useful means for quickly partitioning variant calls for downstream applications. The user is able to specify the genome build that they would like to use, provided it is compatible with ANNOVAR. Finally, using the coding_change function of ANNOVAR and custom code, the mutated amino acid sequence is predicted from annotated nonsynonymous variant calls, and the peptide sequence surrounding the newly introduced amino acid is extracted for epitope prediction. From this step, mutations that give rise to a single amino acid change, and mutations that mutate a larger peptide segment (e.g. indels and stop-losses) are handled separately and reported in separate files to help further assessment.

Once the VCF files have been annotated and partitioned with ANNOVAR, the program determines if HLA haplotypes have been provided by the user containing the HLA-A, -B, and -C haplotypes. NeoPredPipe does not include HLA allele identification as this step in the pipeline is highly dependent upon the source of the data (WES, WGS, targeted gene panels, transcriptome data, or conducted via experimental methods), but the pipeline's github page provides detailed advice on haplotyping from WES/WGS data using the popular tool POLYSOLVER [13], and the output of POLYSOLVER is automatically processed in NeoPredPipe. In cases where no HLA haplotype information is available the most common alleles of each haplotype are assessed; while in cases where the HLA haplotypes are homozygous only that HLA haplotype is used for prediction. HLA haplotypes are cross-referenced with available HLA haplotypes prior to executing netMHCpan ([14]) for the primary neoantigen predictions. As with the primary tool, the user is able to specify the epitope lengths to conduct predictions for (typically epitopes of 8-, 9-, or 10-mers). The output from this process yields a single file containing either filtered or unfiltered (dependent on user options) neoantigen predictions with information on the sample possessing the neoantigen and, in the case of multi-region variant calling, a presence/absence indicator for each of the sequenced regions. These predicted neoantigens are then, optionally, cross-referenced with normal peptides utilizing PeptideMatch ([15]), whereby the candidate epitopes are assessed for novelty against a reference proteome that can be supplied by the user as a fasta file (e.g. from Ensembl or UniProt). When available, users may also provide expression data as a tsv file specific to each sample (or a single reference file) to quickly assess expression levels of the gene carrying a predicted neoantigen. This information is included in the final output table.

The steps outlined above deliver candidate information for neoantigens from provided variant calls that may be presented to cytotoxic T-cells, however, this does not inform the likelihood of a neoantigen eliciting an immune response (i.e. being recognised by a TCR). In order to predict the recognition potential we employ the algorithms and process utilized by [16]. The recognition potential is defined as the product of A and R, where A is the amplitude of the ratio of the relative probabilities of binding for the wild-type and mutant epitopes to the MHC-class I molecules; and R is a measure of similarity to pathogenic peptides, meant to represent the probability that the neoantigen in question is recognised by a TCR clone already present in the tissue/blood. To define A it is necessary to perform neoantigen predictions for the wildtype and mutant epitope: this is not performed by default by NeoPredPipe, but is supplied as an option to employ as a contiguous pipeline. To define R, NeoPredPipe utilizes the multistate thermodynamic model employed by [16], which requires alignment scores for each epitope to a curated Immune Epitope Database list of known epitopes (can be refined and updated by the user, but is provided).