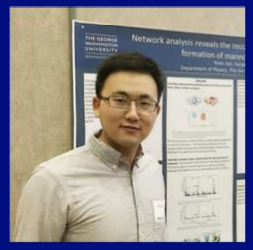
NEC Laboratories America

KDD Washington DC 2022

DARTMOUTH

T-Cell Receptor-Peptide Interaction Prediction with Physical Model Augmented Pseudo-Labeling

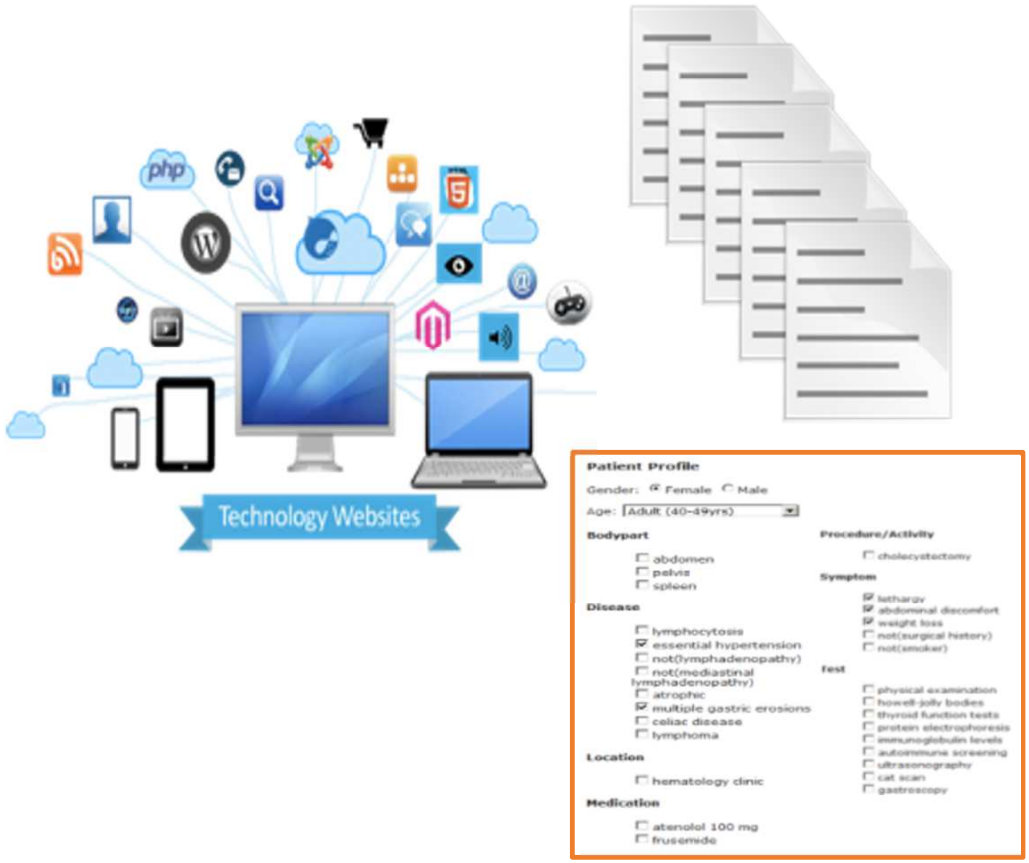
Yiren Jian[1,2] **Erik Kruus[1]** **Martin Renqiang Min[1]**

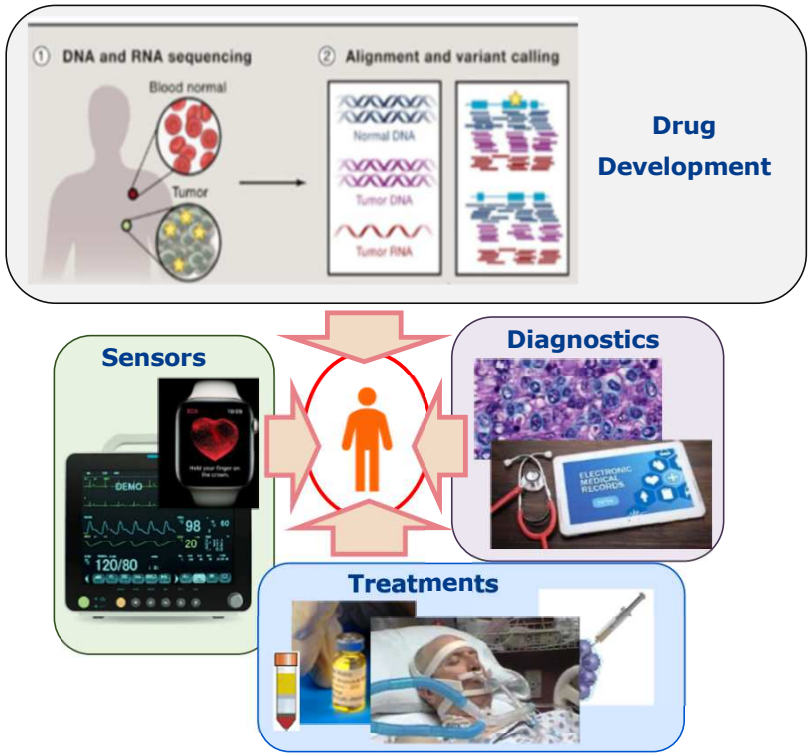


[1]NEC Labs America, [2]Dartmouth College

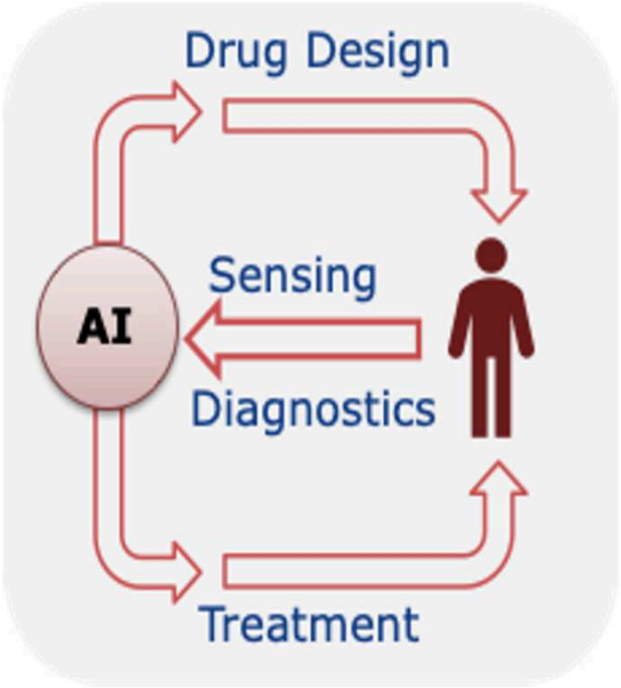
Aug 17, 2022

Background: Modern Healthcare is a Data-Driven Service



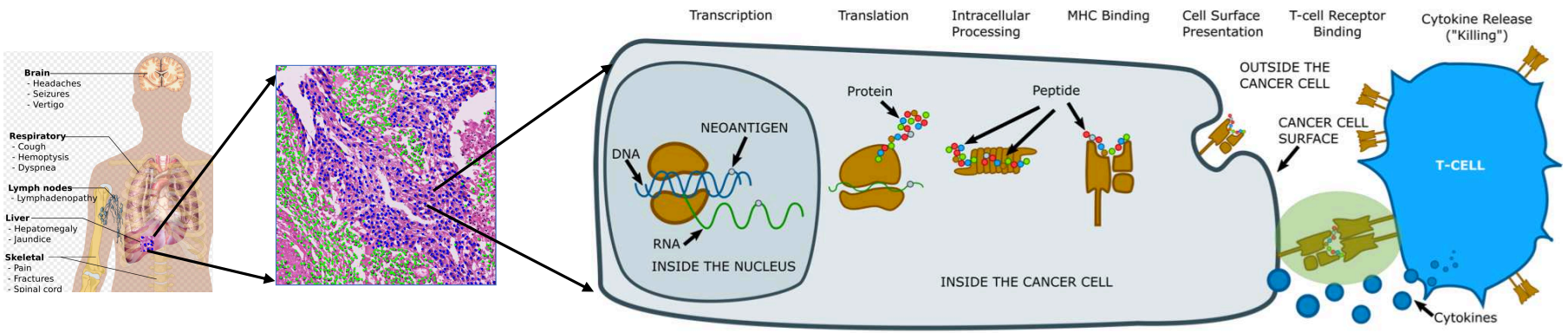


Background: Big Data Enables Precision Medicine

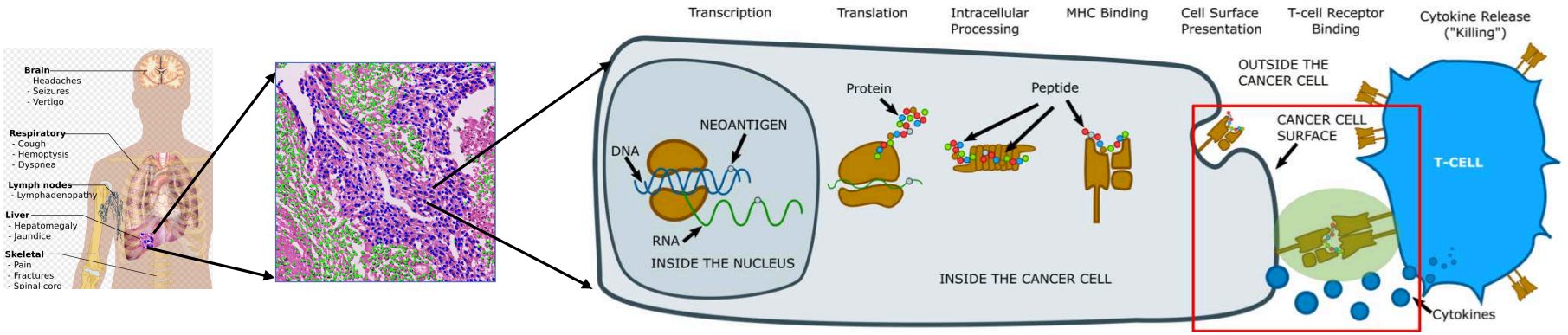


Picture Credit: https://www.nature.com/news/personalized-medicine-time-for-one-person-trials-1.17411

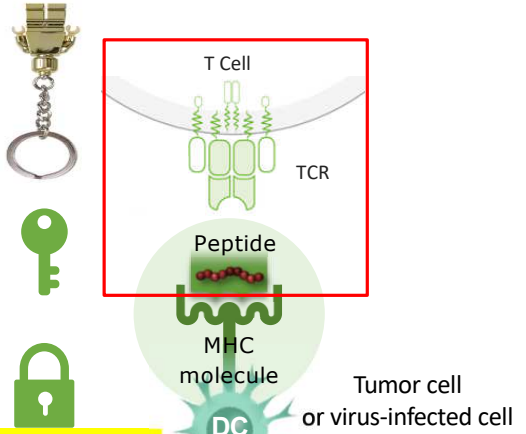
T-Cell Receptor (TCR) Recognition for Precision Immunotherapy



T-Cell Receptor (TCR) Recognition for Precision Immunotherapy

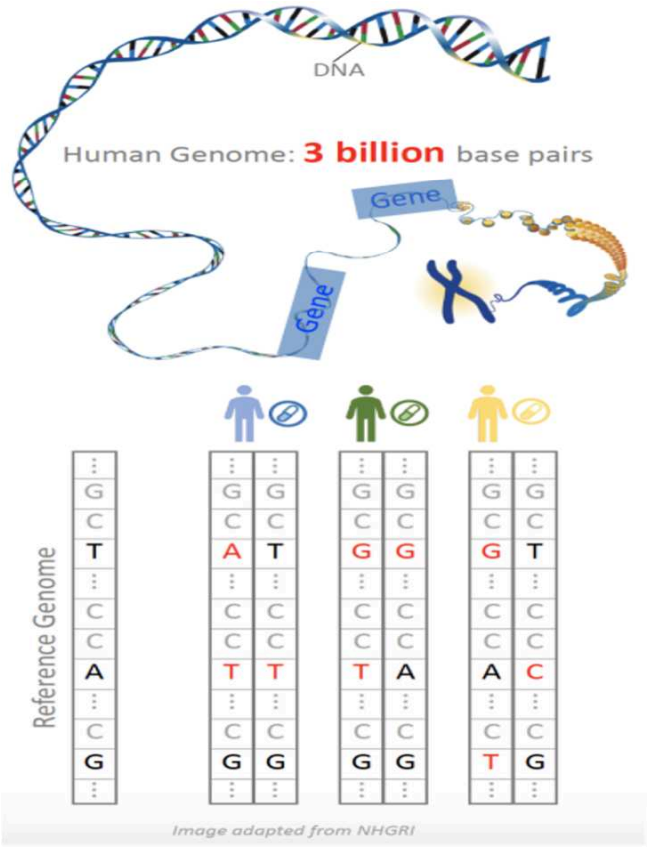


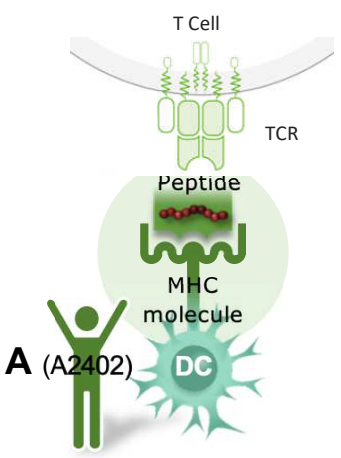
- TCRs bind with **anomalous peptides** (viral / cancerous) presented on the surface of cells
- If TCR-peptide binding takes place, the immune system can **kill cancerous/infected cells**

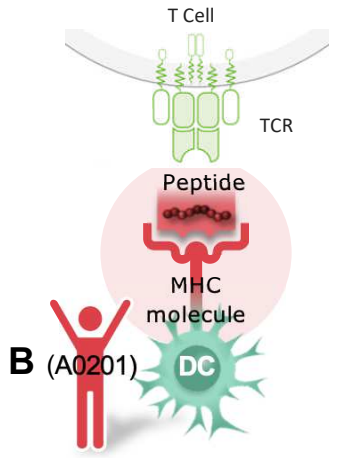


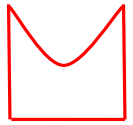
TCR recognition is the holy grail of immunology

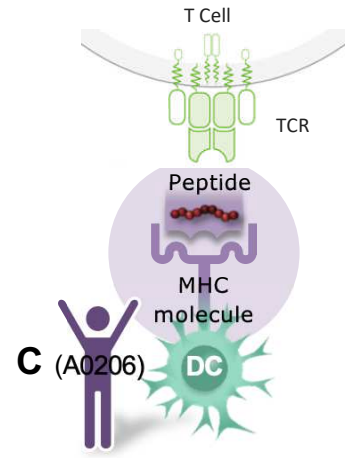
Data-Driven Precision Immunotherapy

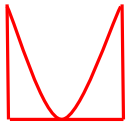






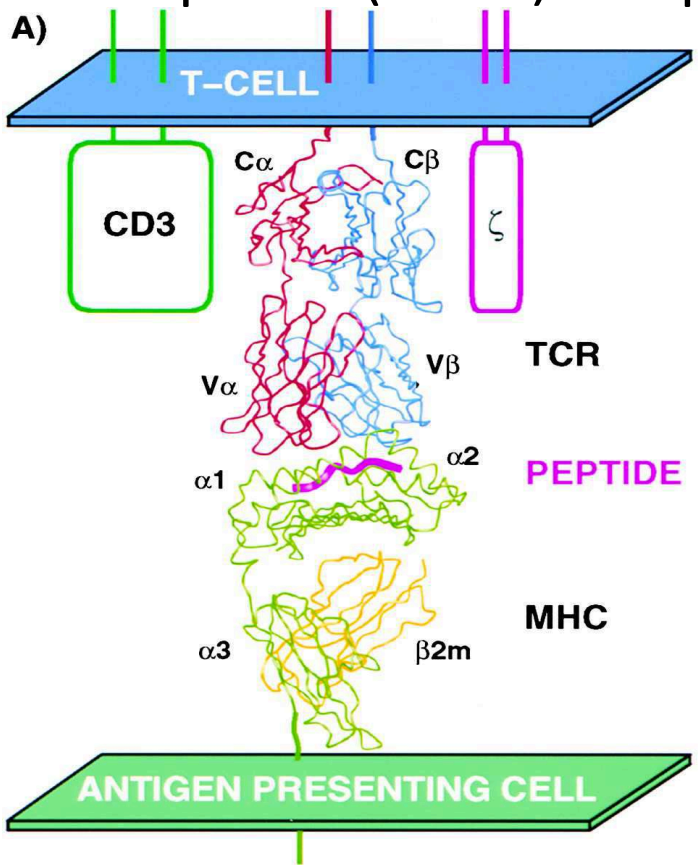


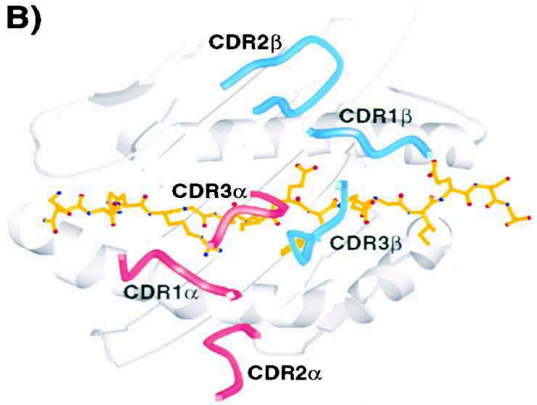


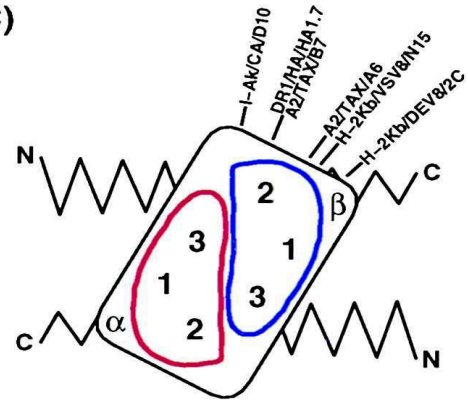


Personalized immunotherapy considers genomic variations of patients

T Cell Receptor (TCR)-Peptide Interaction



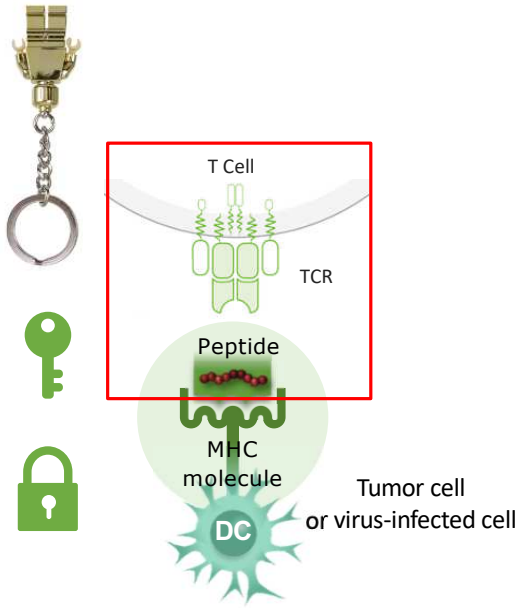




Picture credit: Hennecke and Wiley, Cell 2001.

TCR recognition is the holy grail of immunology

T-Cell Receptor (TCR) Recognition of Peptide Antigens



Our task: Given a sequence of CDR3 beta of **TCR** and a sequence of **peptide**, predict the interaction (binary, 0/1).

For example, for a machine learning model, the inputs are:
TCR: CASSDAGANTEVF and **Peptide**: IKAVYNFATCG
Output is a binary prediction.

TCR recognition is the holy grail of immunology

8

Datasets

McPAS Dataset[1]

CDR3.beta.aa	Species	Category	Protein.ID	Epitope.peptide
All	All	All	/	All
CASSDAGANTEVF	Mouse	Pathogens	P09991	IKAVYNFATCG
CASSDAGAYAEQF	Mouse	Pathogens	P09991	IKAVYNFATCG
CASSDAGGAAEVF	Mouse	Pathogens	P09991	IKAVYNFATCG
CASSDAGHSPLYF	Mouse	Pathogens	P09991	IKAVYNFATCG

VDJdb Dataset[2]

CASSYLPGQGDHYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEKGGL
CASSFEAGQGFFSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEKGGL
CASSFEPGQGFYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEKGGL
CASSYEPGQVSHYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEKGGL
CASSALASLNEQFF	TRBV14	TRBJ2-1	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEKGGL
CASSYLPGQGDHYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEQGGL
CASSFEAGQGFFSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEQGGL
CASSFEPGQGFYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEQGGL
CASSYEPGQVSHYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKEQGGL
CASSYLPGQGDHYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKETGGL
CASSFEAGQGFFSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKETGGL
CASSFEPGQGFYSNQPQHF	TRBV13	TRBJ1-5	HomoSapiens	HLA-B*08	B2M	MHCI	FLKETGGL

[1] McPAS-TCR: a manually curated catalogue of pathology-associated T cell receptor sequences, Bioinformatics 2017
[2] VDJdb in 2019: database extension, new analysis infrastructure and a T-cell receptor motif compendium, Nucleic Acids Research 2020

Challenges on Learning Deep Models

1. The dataset are relatively small: 20,000 TCRs in McPAS and 40,000 in VDJdb.

2. The examples in datasets are biased, i.e., there are only 200 and 300 different peptides in McPAS and VDJdb.

Potential Solutions

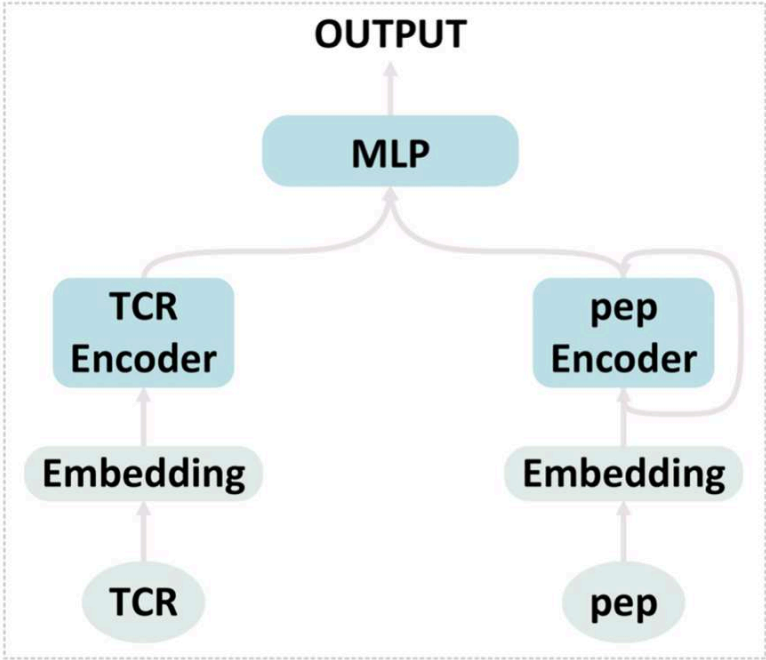
1. Machine Learning based data-augmentation: Pseudo-labeling and Self-training.

2. Data-augmentation by physical simulation, e.g., leveraging fast computing of binding energies by physical computation (e.g., Docking).

Especially, there exist millions of TCR sequences in database, like TCRdb[1].

[1] TCRdb: a comprehensive database for T-cell receptor sequences with powerful search function, Nucleic Acids Research 2022

Our Model

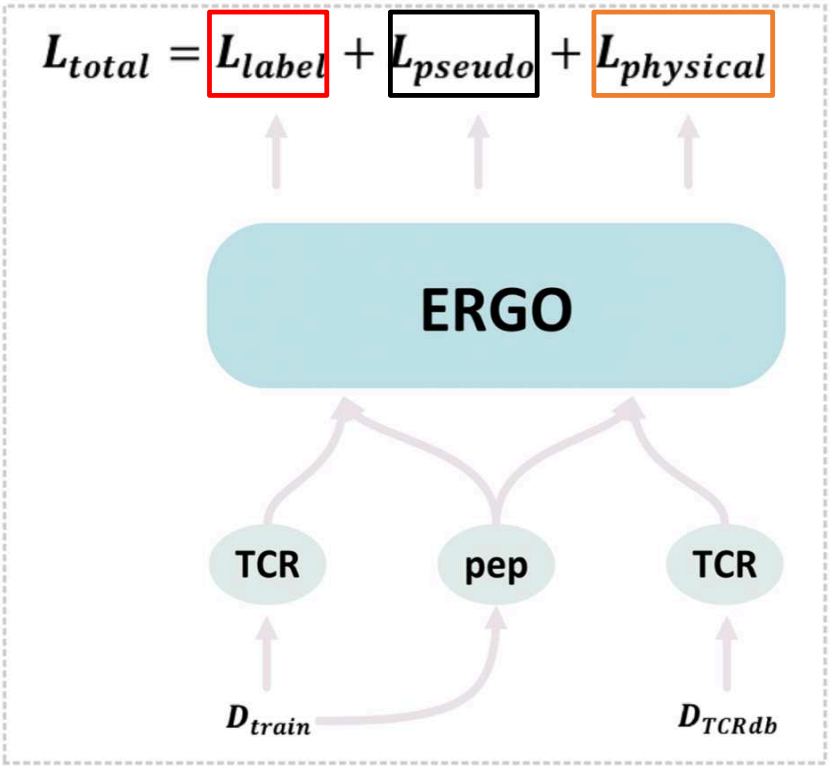


- **Our contribution is introducing a universal learning framework with pseudo labeling**.
  - we adapt a simple base model from ERGO[1], which has one encoder for TCR and another one for peptide.

- Our learning framework should be agnostic to different base models.

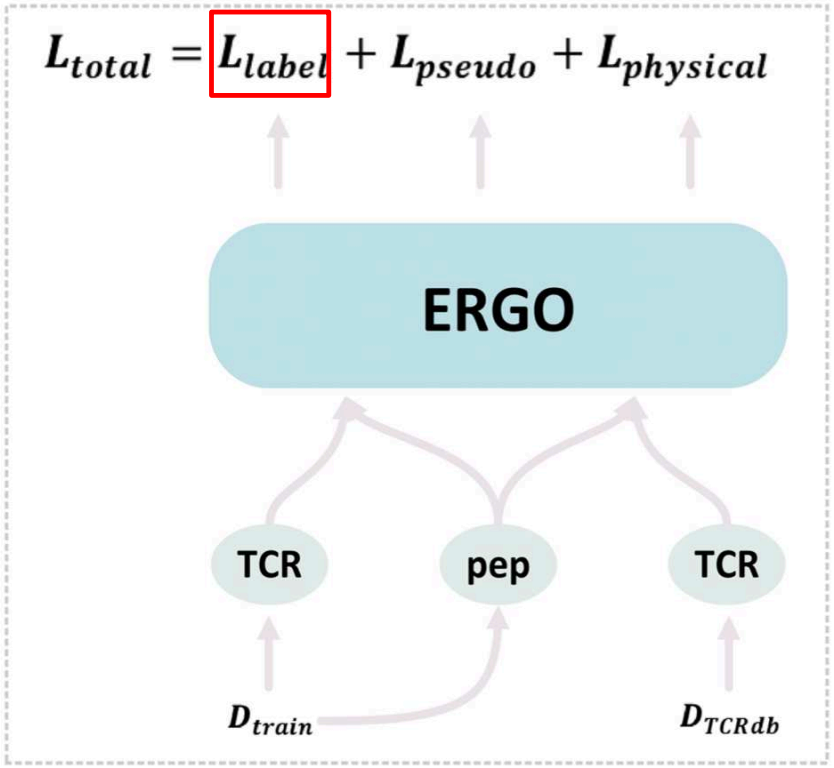
[1] Prediction of Specific TCR-Peptide Binding From Large Dictionaries of TCR-Peptide Pairs. Front. Immunol. 2020

Our Learning Losses



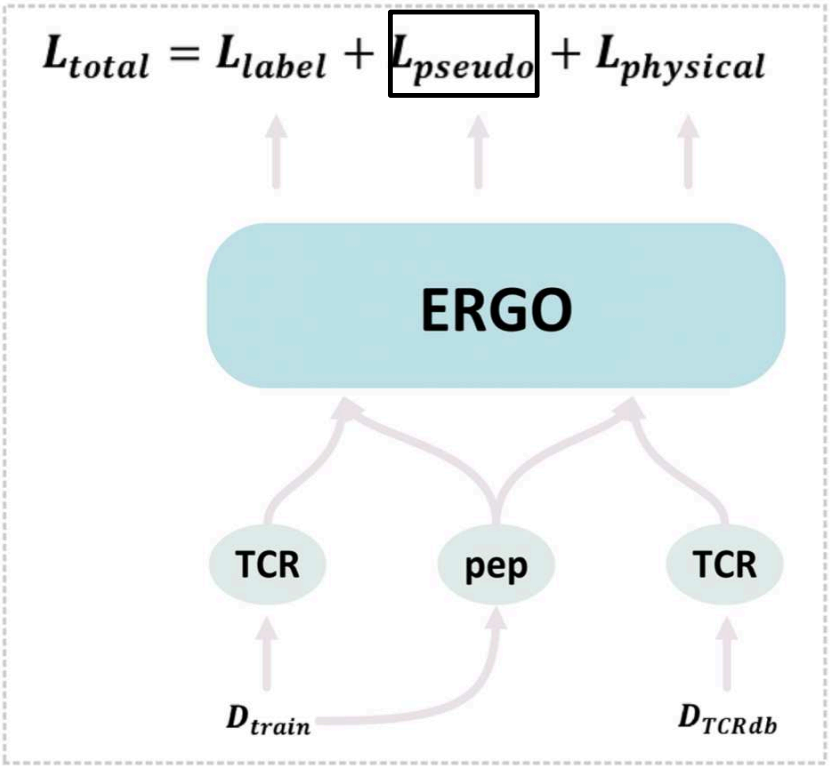
1. Supervised learning from limited labeled data.
2. Learning from pseudo-labeled data (by a teacher model pretrained).
3. Learning from data with surrogate labels by physical modeling.

Our Learning Losses



This is standard supervised learning from data from datasets like McPAS or VDJdb.

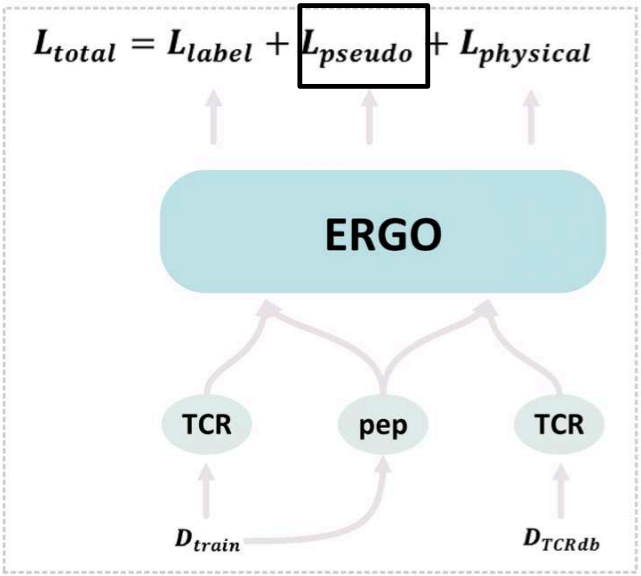
Our Learning Losses



Suppose we have a model pre-trained on the McPAS and VDJdb, we use this model to pseudo-label TCR and peptide pairs from **external** TCR database.

For example, *TCR:CASSFRGSETQYF* and *Peptide: IKAVYNFATCG* are labeled by 0.82

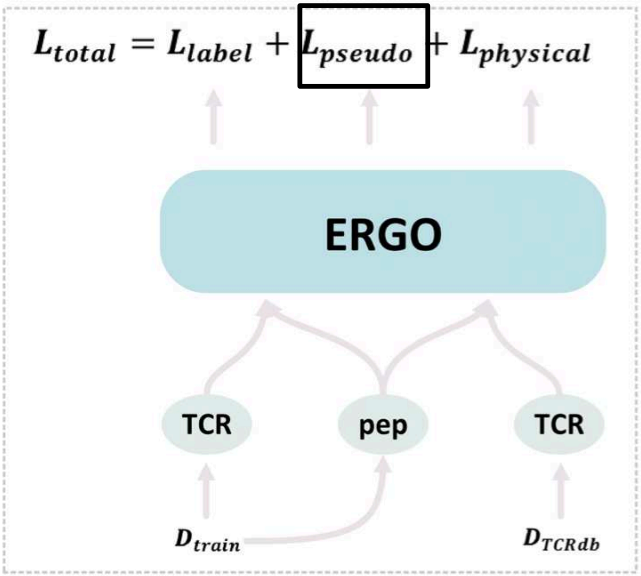
Why Pseudo Labeling Works



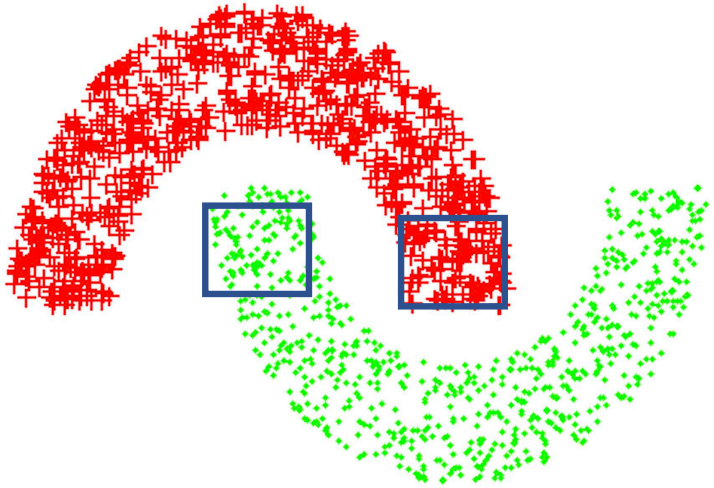
- The decision boundary should lie in low-density regions in order to improve generalization
- Unlabeled samples that lie either near or far from labeled samples should be informative for decision boundary estimation.
- Pseudo-labeling generally works by iteratively propagating labels from labeled samples to unlabeled samples using the current model to relabel the data.[1]

[1] Cascante-Bonilla et al, Curriculum Labeling: Revisiting Pseudo-Labeling for Semi-Supervised Learning, AAAI 2021

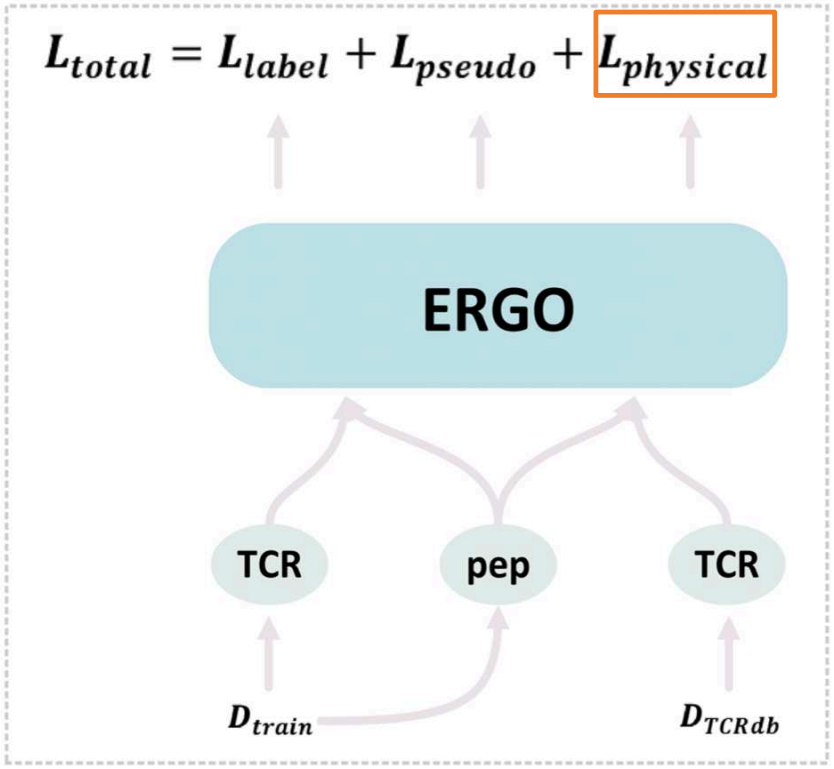
Pseudo Labeling is Not a Silver Bullet



- The key lies in the choice of the unlabeled data for pseudo labeling

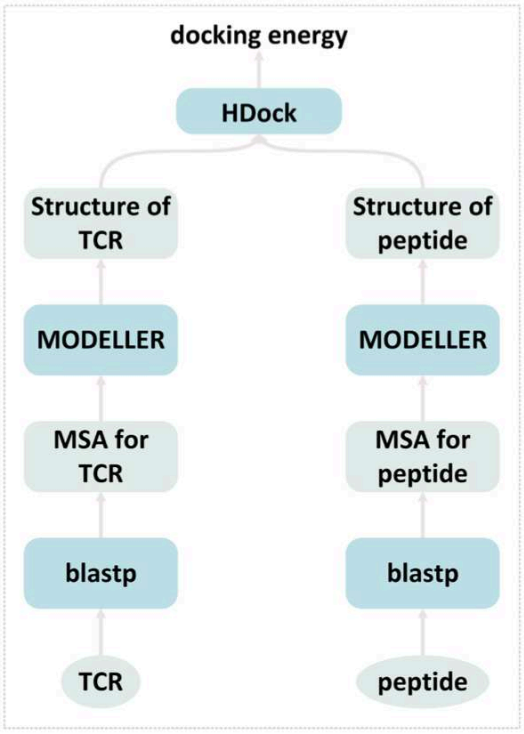


Physical Augmentation by Docking for Pseudo Labeling



We compute **Docking energy** between a TCR and peptide, as the surrogate label, to extend our dataset for pseudo labeling.

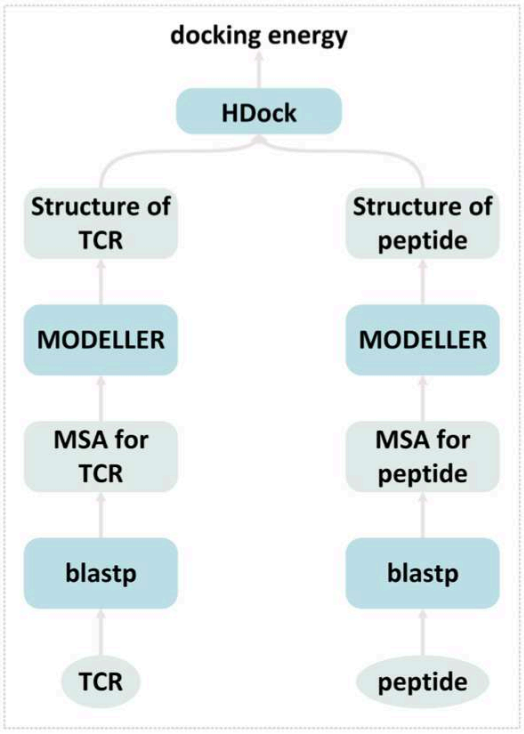
Docking by HDock[1]



Docking is a computational method for predicting the structures of protein complex by minimizing an **energy scoring function**.

[1] The HDOCK server for integrated protein–protein docking, Nature Protocols 2020

Docking by HDock[1]

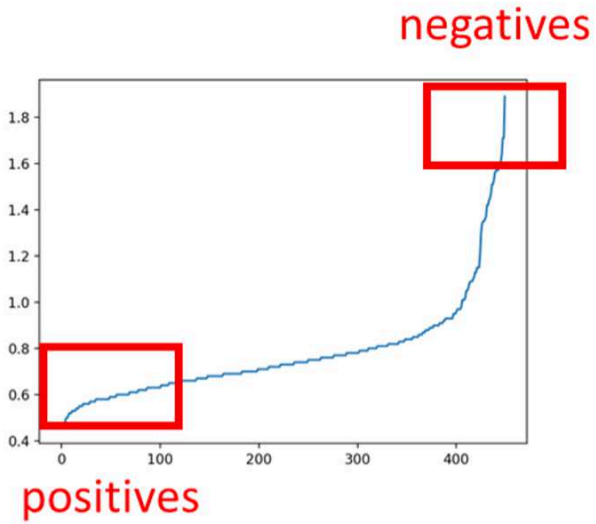


1. For sequences without 3D structures, we run **blastp** for multiple sequence alignment (MSA) to find homologous sequence with known structures.

2. Then, we use MODELLER to build 3D structures of sequences (both TCRs and peptides).

[1] The HDOCK server for integrated protein–protein docking, Nature Protocols 2020

Docking by HDock[1]



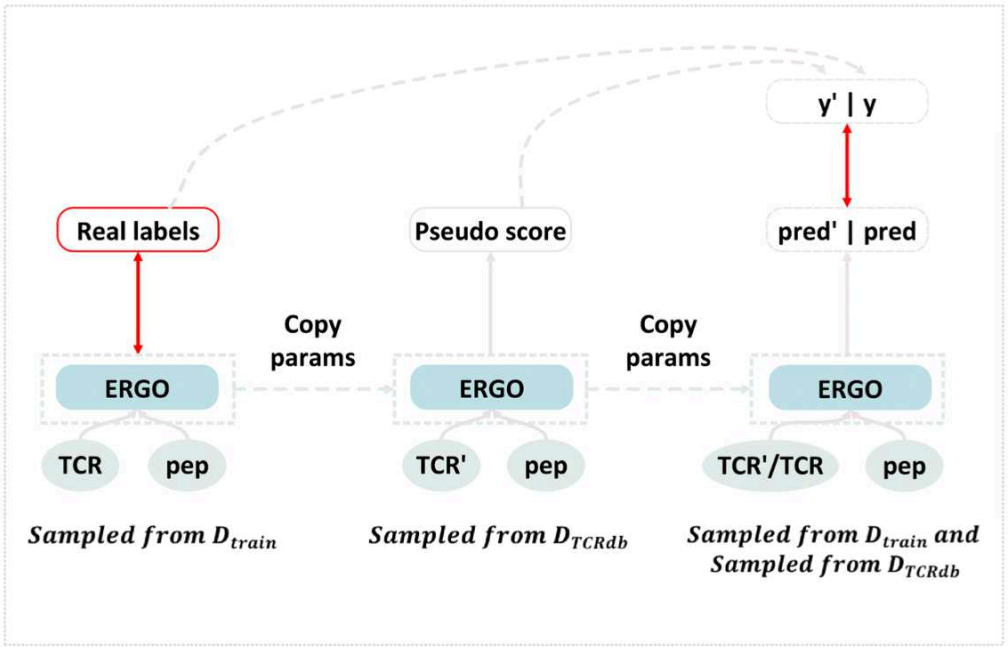
Energy distribution at final

1. We use the minimal energies found during the docking optimization as surrogate labels.

2. Pairs with the least 25 percentile energies are used as positive pairs and top 25 percentile are used as negatives.

[1] The HDOCK server for integrated protein–protein docking, Nature Protocols 2020

Our Proposed Method



$$\mathcal{L}_{labeled} = \text{BinaryCrossEntropy}(pred, y)$$ ← Supervised loss from labeled TCR-peptides

$$\mathcal{L}_{\text{physical}} = \text{BinaryCrossEntropy}(pred', y')$$ ← Learning from pseudo-labels by docking

$$\mathcal{L}_{\text{pseudo-labeled}} = \text{KL-div}(pred', prob')$$ ← Learning from pseudo-labels by a teacher

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{labeled}} + \beta\mathcal{L}_{\text{physical}} + \gamma\mathcal{L}_{\text{pseudo-labeled}}$$

Our Algorithm

$(t, p), y = Sample(Dataset_{\text{train}})$
$(t', p'), y' = Sample(Dataset_{\text{auxiliary}})$
▷ **Learning from labeled dataset**
$output_1 = model(t, p)$
$L_{labeled} = BCE(output_1, y)$
$L_{labeled}.backward()$
$optimizer.step()$
▷ **Learning from data-augmented pseudo-labeling**
$output_2 = model(t', p')$
$output'_2 = teacher\_model(t', p')$
$L_{pseudo-labeled} = KL(output_2, output'_2)$
$L_{pseudo-labeled}.backward()$
$optimizer.step()$
$params = model.parameters()$
▷ **Learning from physical modeling**
$output_3 = model(t', p')$
$L_{physical} = BCE(output_3, y')$
$L_{physical}.backward()$
$optimizer.step()$
▷ **Look ahead meta-update**
$optimizer.learning\_rate \times 2$
$output_4 = model(t, p)$
$L'_{labeled} = BCE(output_4, y)$
if $L'_{labeled} > L_{labeled}$ **then**
  $model.load(params)$
end if

1. Learning from labeled dataset
2. Learning from data-augmented pseudo-labeling
3. Learning from physical modeling
4. Look ahead meta-update

Our Algorithm

$(t, p), y = Sample(Dataset_{\text{train}})$
$(t', p'), y' = Sample(Dataset_{\text{auxiliary}})$
▷ **Learning from labeled dataset**
$output_1 = model(t, p)$
$L_{labeled} = BCE(output_1, y)$
$L_{labeled}.backward()$
$optimizer.step()$
▷ **Learning from data-augmented pseudo-labeling**
$output_2 = model(t', p')$
$output'_2 = teacher\_model(t', p')$
$L_{pseudo-labeled} = KL(output_2, output'_2)$
$L_{pseudo-labeled}.backward()$
$optimizer.step()$
$params = model.parameters()$
▷ **Learning from physical modeling**
$output_3 = model(t', p')$
$L_{physical} = BCE(output_3, y')$
$L_{physical}.backward()$
$optimizer.step()$
▷ **Look ahead meta-update**
$optimizer.learning\_rate \times 2$
$output_4 = model(t, p)$
$L'_{labeled} = BCE(output_4, y)$
if $L'_{labeled} > L_{labeled}$ **then**
  $model.load(params)$
end if

1. Learning from labeled dataset
2. Learning from data-augmented pseudo-labeling
3. Learning from physical modeling
4. Look ahead meta-update

Our Algorithm

$(t, p), y = Sample(Dataset_{\text{train}})$
$(t', p'), y' = Sample(Dataset_{\text{auxiliary}})$
▷ **Learning from labeled dataset**
$output_1 = model(t, p)$
$L_{labeled} = BCE(output_1, y)$
$L_{labeled}.backward()$
$optimizer.step()$
▷ **Learning from data-augmented pseudo-labeling**
$output_2 = model(t', p')$
$output'_2 = teacher\_model(t', p')$
$L_{pseudo-labeled} = KL(output_2, output'_2)$
$L_{pseudo-labeled}.backward()$
$optimizer.step()$
$params = model.parameters()$
▷ **Learning from physical modeling**
$output_3 = model(t', p')$
$L_{physical} = BCE(output_3, y')$
$L_{physical}.backward()$
$optimizer.step()$
▷ **Look ahead meta-update**
$optimizer.learning\_rate \times 2$
$output_4 = model(t, p)$
$L'_{labeled} = BCE(output_4, y)$
if $L'_{labeled} > L_{labeled}$ **then**
  $model.load(params)$
end if

1. Learning from labeled dataset
2. Learning from data-augmented pseudo-labeling
3. Learning from physical modeling
4. Look ahead meta-update

Our Algorithm

$(t, p), y = Sample(Dataset_{\text{train}})$
$(t', p'), y' = Sample(Dataset_{\text{auxiliary}})$
▷ **Learning from labeled dataset**
$output_1 = model(t, p)$
$L_{labeled} = BCE(output_1, y)$
$L_{labeled}.backward()$
$optimizer.step()$
▷ **Learning from data-augmented pseudo-labeling**
$output_2 = model(t', p')$
$output'_2 = teacher\_model(t', p')$
$L_{pseudo-labeled} = KL(output_2, output'_2)$
$L_{pseudo-labeled}.backward()$
$optimizer.step()$
$params = model.parameters()$
▷ **Learning from physical modeling**
$output_3 = model(t', p')$
$L_{physical} = BCE(output_3, y')$
$L_{physical}.backward()$
$optimizer.step()$
▷ **Look ahead meta-update**
$optimizer.learning\_rate \times 2$
$output_4 = model(t, p)$
$L'_{labeled} = BCE(output_4, y)$
if $L'_{labeled} > L_{labeled}$ **then**
  $model.load(params)$
end if

1. Learning from labeled dataset
2. Learning from data-augmented pseudo-labeling
3. Learning from physical modeling
4. Look ahead meta-update

We will only update parameters of the model if and only if the learning from surrogate labels reduces the loss (on labeled dataset).

Experiments on McPAS Dataset

Data size	6K	10K	20K
ERGO	54.4 ± 0.5	56.3 ± 0.5	71.2± 0.3
+ Pseudo	58.5 ± 0.5	62.7 ± 0.4	72.7± 0.3
+ Docking	61.4 ± 0.4	64.8 ± 0.4	72.4± 0.4
ours (3 losses)	62.1 ± 0.4	66.0 ± 0.4	73.2± 0.3
ours + meta-update	**63.4± 0.4**	**66.5± 0.4**	**74.2± 0.3**

Experimental results with ERGO-AE (using auto-encoder for TCR and a double-LSTM for Peptides).

Experiments on McPAS Dataset

Data size	6K	10K	20K
ERGO	67.6 ± 0.4	71.9 ± 0.4	76.6 ± 0.3
+ Pseudo	69.3 ± 0.4	73.6 ± 0.3	77.6 ± 0.3
+ Docking	69.4 ± 0.4	73.3 ± 0.3	77.9 ± 0.2
ours (3 losses)	70.4 ± 0.3	73.7 ± 0.3	77.6 ± 0.2
ours + meta-update	**71.5 ± 0.3**	**74.7 ± 0.3**	**78.4±0.2**

Experimental results with ERGO-LSTM (using both LSTMs for TCRs and Peptides).

Experiments on VDJdb Dataset

Data size	6K	10K	20K
ERGO	60.7 ± 0.5	61.0 ± 0.5	66.8 ± 0.4
+ Pseudo	61.0 ± 0.5	63.9 ± 0.4	69.8± 0.3
+ Docking	62.2 ± 0.5	64.6 ± 0.5	71.5 ± 0.3
ours (3 losses)	63.4 ± 0.5	66.4 ± 0.4	72.2 ± 0.3
ours + meta-update	**64.6 ± 0.5**	**67.6 ± 0.4**	**72.9 ± 0.3**

Experimental results with ERGO-AE (using auto-encoder for TCR and a double-LSTM for Peptides).

Experiments on VDJdb Dataset

Data size	6K	10K	20K
ERGO	68.1± 0.4	72.0 ± 0.3	73.6 ± 0.4
+ Pseudo	68.4 ± 0.3	72.4 ± 0.3	73.9 ± 0.3
+ Docking	69.5 ± 0.4	73.4± 0.3	74.6± 0.3
ours (3 losses)	70.4± 0.3	72.9± 0.3	74.6± 0.3
ours + meta-update	**71.5 ± 0.3**	**73.8± 0.3**	**75.2± 0.3**

Experimental results with ERGO-LSTM (using both LSTMs for TCRs and Peptides).

Analysis on Rare Peptides

rare peptides	baseline	average	ours
KRWIILGLNK	52.8	54.4	68.1
KMVAVFYTT	48.9	54.4	65.8
FPRPWLHGL	50.2	54.4	58.5

Experiments with AE-LSTM model with McPAS dataset of 6K labeled examples
"average" denotes the average AUC for all peptides in this experimental setup

Conclusions

- We propose a general method for training a deep learning model with physical modeling augmented pseudo-labeling:

  1. Physical modeling between TCRs and peptides by docking to select what kind of unlabeled data for pseudo labeling, leveraging domain knowledge and inductive biases
  2. Data-augmented pseudo-labeling of TCR-peptide pairs with a model first trained on the labeled dataset.
  3. Look-ahead meta-update to further remove noise in physical modeling

- We introduce a new dataset that contains over 80,000 unknown TCR-peptide pairs with docking energy scores.

Future Research

- ❑ Population-wide TCR characterization and generalization of TCR-peptide interaction prediction
- ❑ Incorporate MHC Class I/II sequence/ structure information into TCR-peptide-MHC interaction prediction

"Deep learning has instead given us machines with truly impressive abilities but no intelligence. The difference is profound and lies in the absence of a model of reality."

— **Judea Pearl,** **The Book of Why: The New Science of Cause and Effect**

Studying the synergy between physical modeling and data-driven deep learning

Acknowledgement

NEC Labs America: Alexandru Niculescu, Eric Cosatto, Kai Li, Hans Peter Graf

Canada National Research Council: Hongyu Guo

NEC OncoImmunity: Trevor Clancy

NEC Labs Europe: Filippo Grazioli, Pierre Machart, Anja Moesch

Ohio State University: Ziqi Chen, Xia Ning

Yale University: Murilo Dorion, Prashant Emani, Tianxiao Li, Mark Gerstein

University of Hong Kong: Tung-Chi IP, Jun Yang

Questions?

\Orchestrating a brighter world

NEC

https://www.nec-labs.com/research/machine-learning/