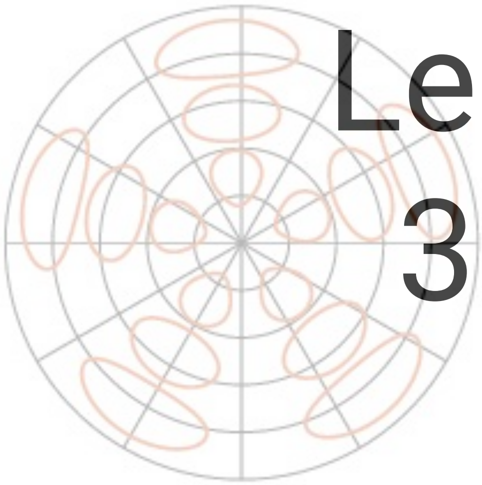
Class core values

1. Be **respect**ful to yourself and others
2. Be **confident** and believe in yourself
3. Always do your **best**
4. Be **cooperative**
5. Be **creative**
6. Have **fun**
7. Be **patient** with yourself while you learn
8. Don’t be shy to **ask “stupid” questions**
9. Be **inclusive** and **accepting**



Week 9, Lecture 1

Learning on 3D objects

Learning Objectives

1. Explain the challenges of 3D objects
2. Describe the basic concepts of main methods for learning on 3D objects
3. Apply hyperparameter tuning to learning progblems

Types of input data for proteins

1. Simple input SVM, random forest, dense neural net

protein$_1$	25 kDa	pI=7.5	310 residues	…	2.5 hr half-life	Stability$_1$
protein$_2$	10 kDa	pI=4	50 residues	…	10 hr half-life	Stability$_2$
protein$_3$	100 kDa	pI=8	1200 residues	…	2 hr half-life	Stability$_3$

…

Types of input data for proteins

1. Simple input — SVM, Random Forest, dense neural net
2. 2D image — CNN

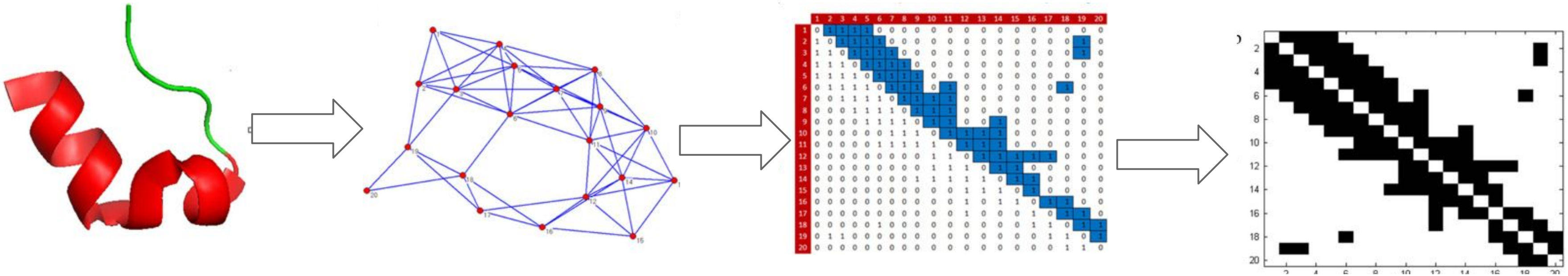


Image from *Physcia A*, **2017**, volume 456, 782-791

Types of input data for proteins

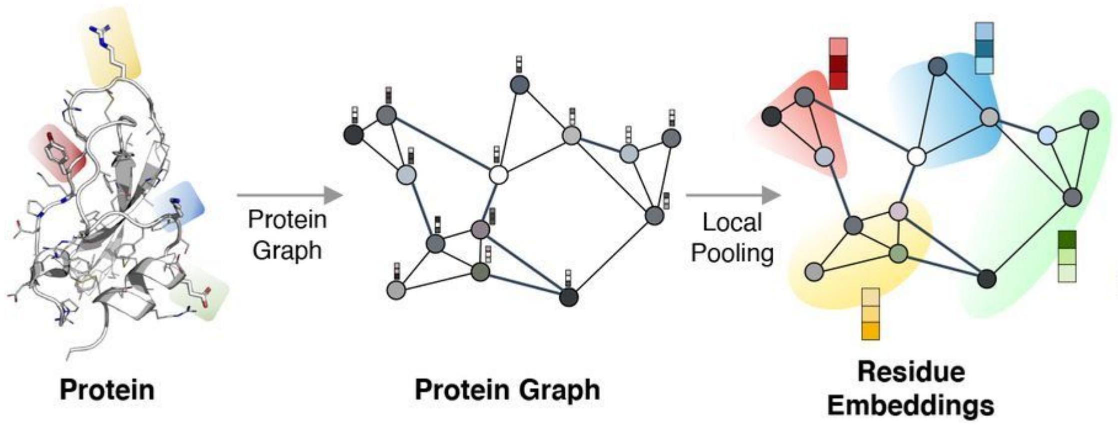
1. Simple input — SVM, Random Forest, dense neural net
2. 2D image — CNN
3. String of amino acids — Natural language processing

p_1	MGLTDILGFNREFDILAV…SPLFG	s_1
p_2	MLKPTRVNMSERCGHITDENVCSR…TLVRF	s_2
p_3	MIKRTVIHGRDFRWNYTSPL…GMNSWQ	s_3
…		

Features: charge, pKa, size, functional groups, hydrogen bond status, …

Types of input data for proteins

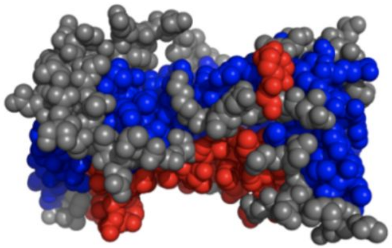
1. Simple input — SVM, Random Forest, dense neural net
2. 2D image — Convolutional neural nets
3. String of amino acids — Natural language processing (RNN, LSTM, Transformers)
4. Graphs — Graph Convolutional Neural nets



Types of input data for proteins

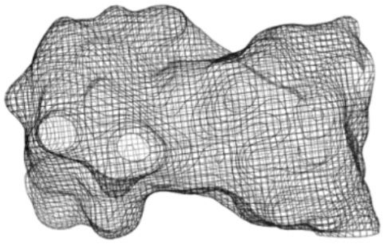
1.	Simple input	SVM, Random Forest, dense neural net
2.	2D image	Convolutional neural nets
3.	String of amino acids	Natural language processing (RNN, LSTM, Transformers)
4.	Graphs	Graph convolutional neural nets
5.	3D objects	

Set of balls / Point cloud



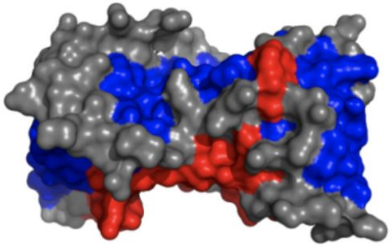
classical statistical potentials; Eismann et al. 2020

Gaussian clouds



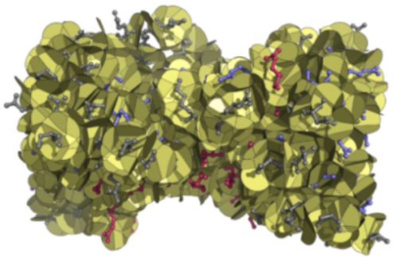
Derevyanko et al. Bioinformatics 2018; Pages et al. Bioinformatics 2019;

Molecular surface



Olechnovic & Venclovas, Proteins 2017; Correia, Bronstein et al. Nat Met 2020

3D tessellation



Igashov et al. Bioinformatics 2021; Olechnovic et al. Proteins 2021

Slide from Sergei Grudinin, AlgoSB 2021

We live in a 3D world and obtaining 3D data is becoming increasingly common

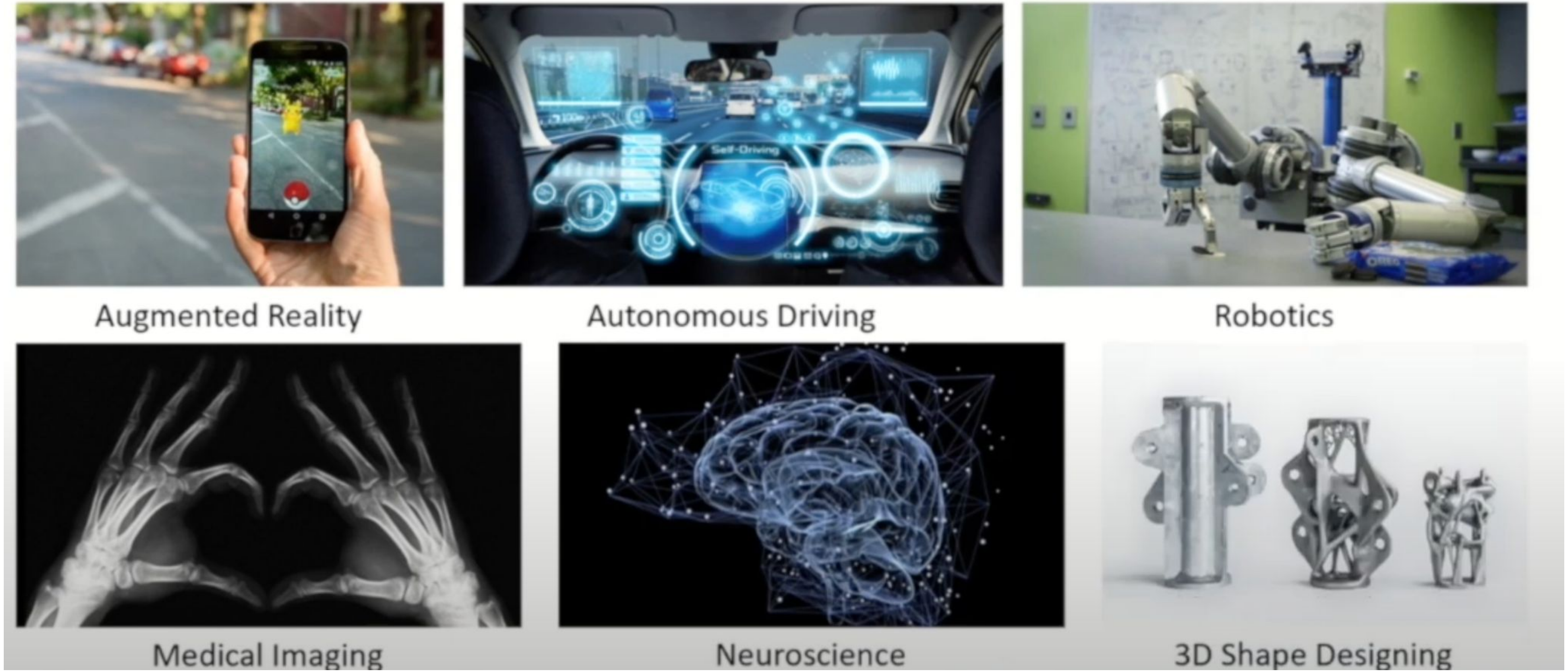
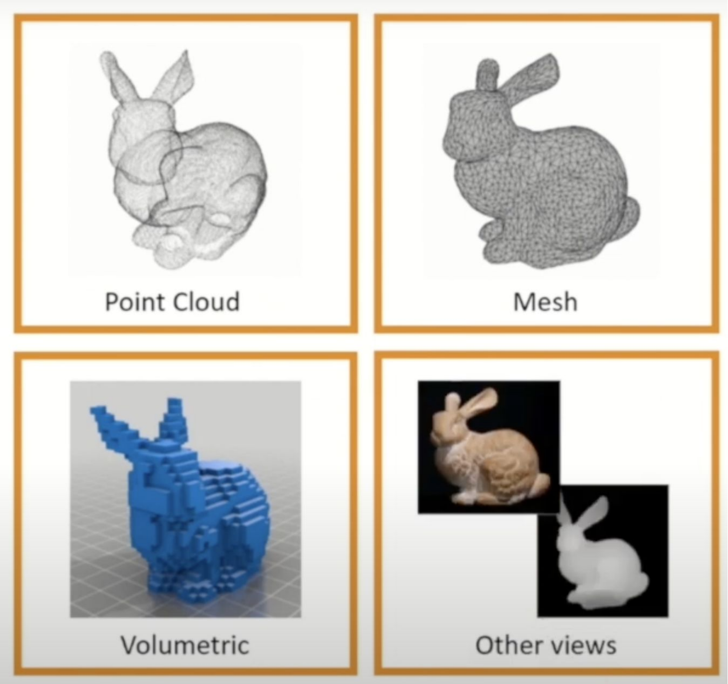
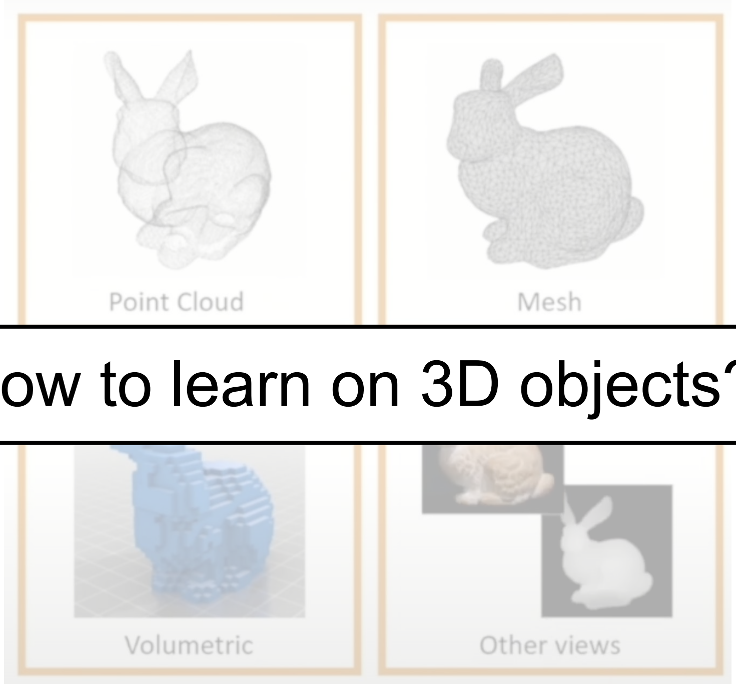


Image from Shivansh Mittal, PointNet

We live in a 3D world and obtaining 3D data is becoming increasingly common



We live in a 3D world and obtaining 3D data is becoming increasingly common

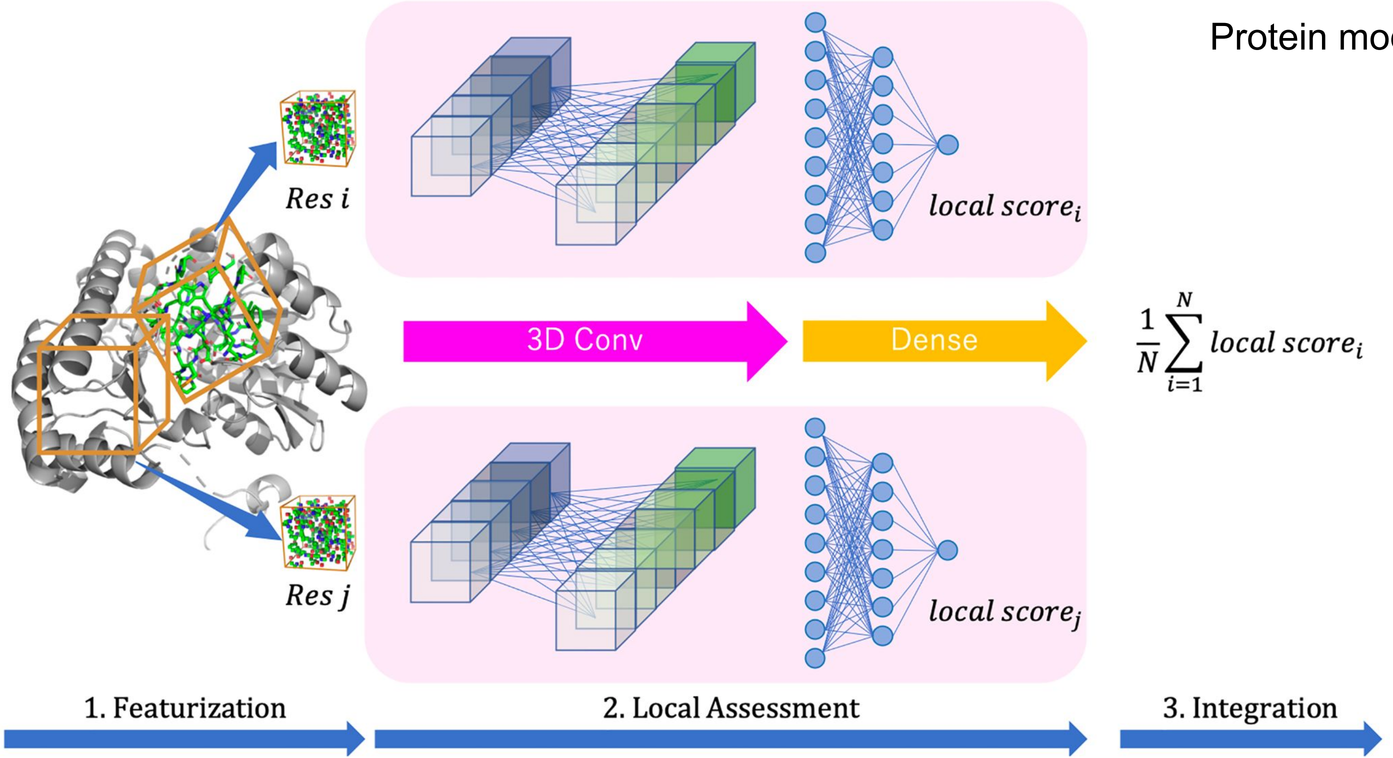


How to learn on 3D objects?

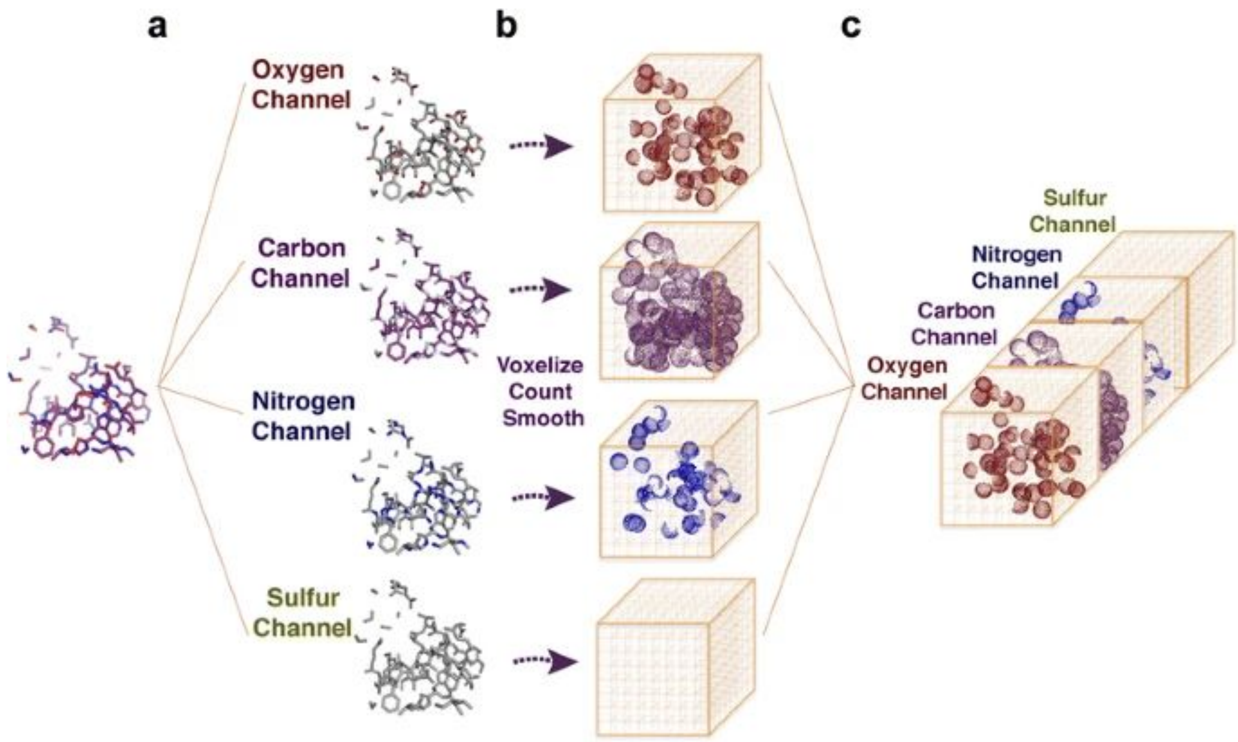
Method 1. Gridding the surface and using CNN

Method 1. Gridding the surface and using CNN

Protein model accuracy estimation



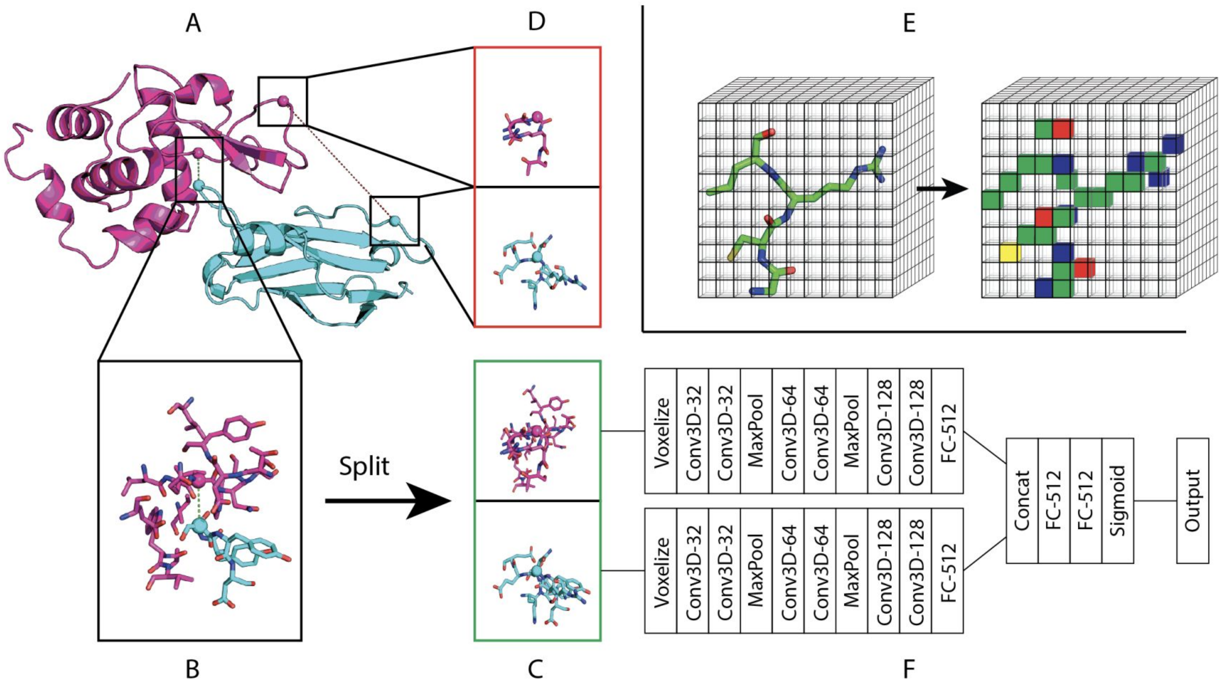
Method 1. Gridding the surface and using CNN



Amino acid environment similarity analysis

Image from Torng and Altman, *BMC Bioinformatics* **2017**

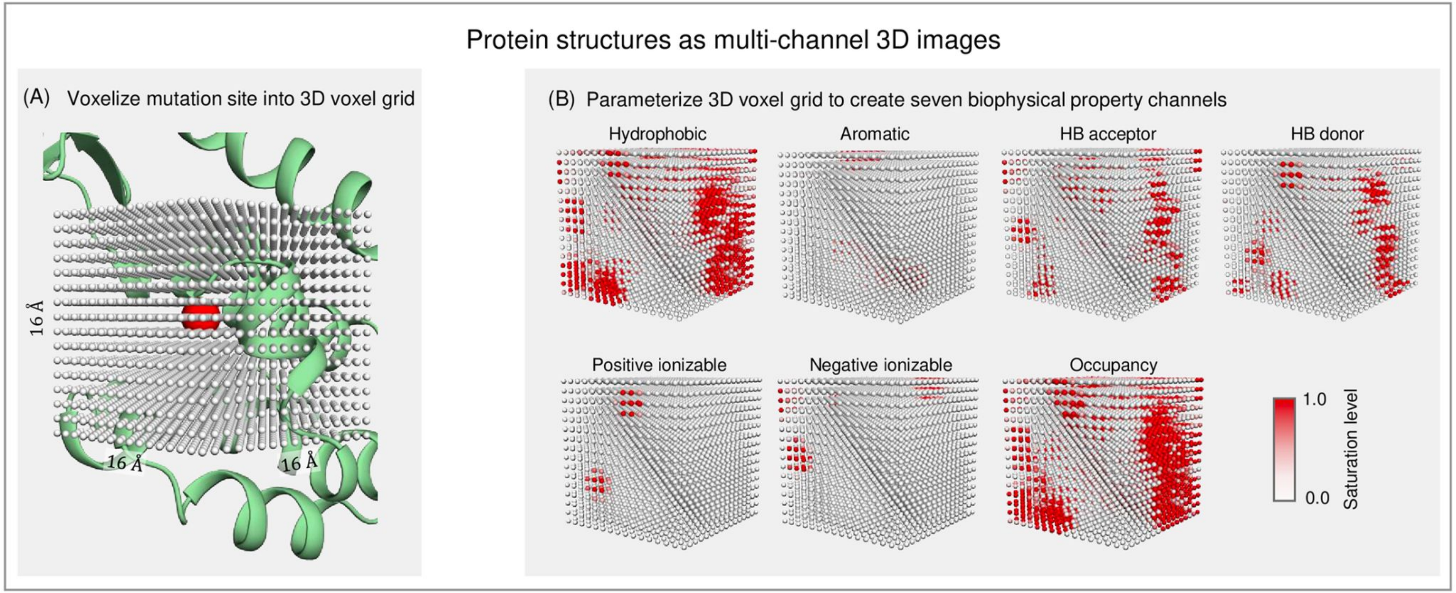
Method 1. Gridding the surface and using CNN



Interface prediction

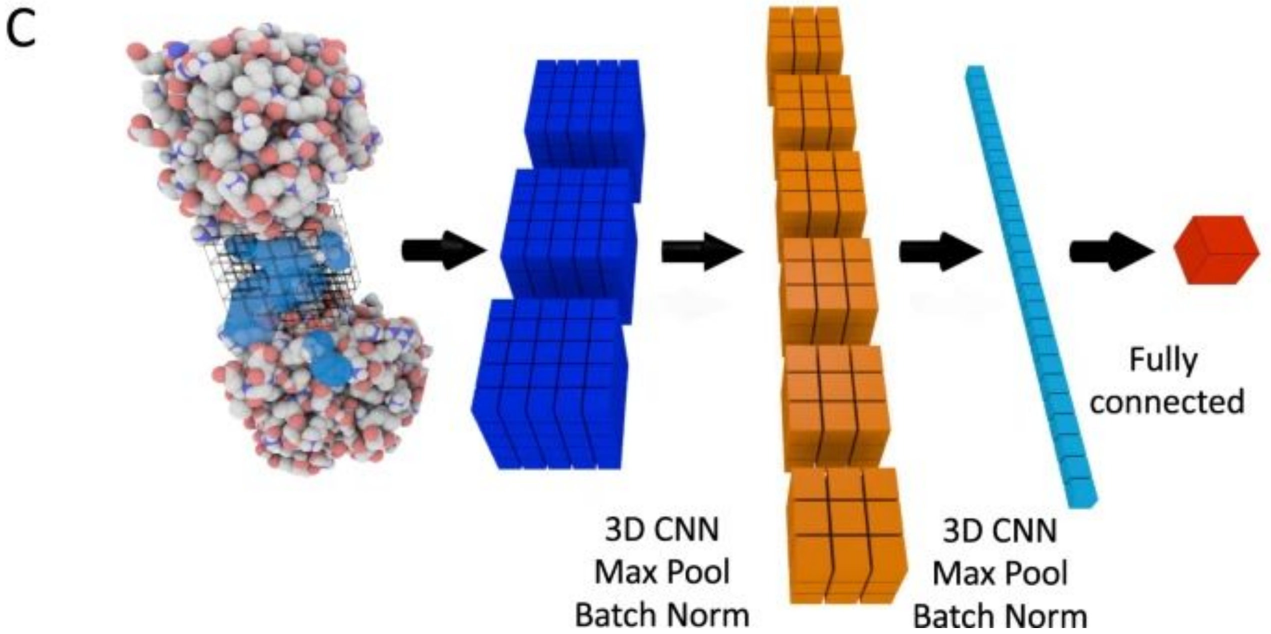
Method 1. Gridding the surface and using CNN

Changes in stability upon point mutation



Method 1. Gridding the surface and using CNN

Data mining 3D protein-protein interfaces



Method 2. Tessellation combined with GNN

Method 2. Tessellation combined with GNN

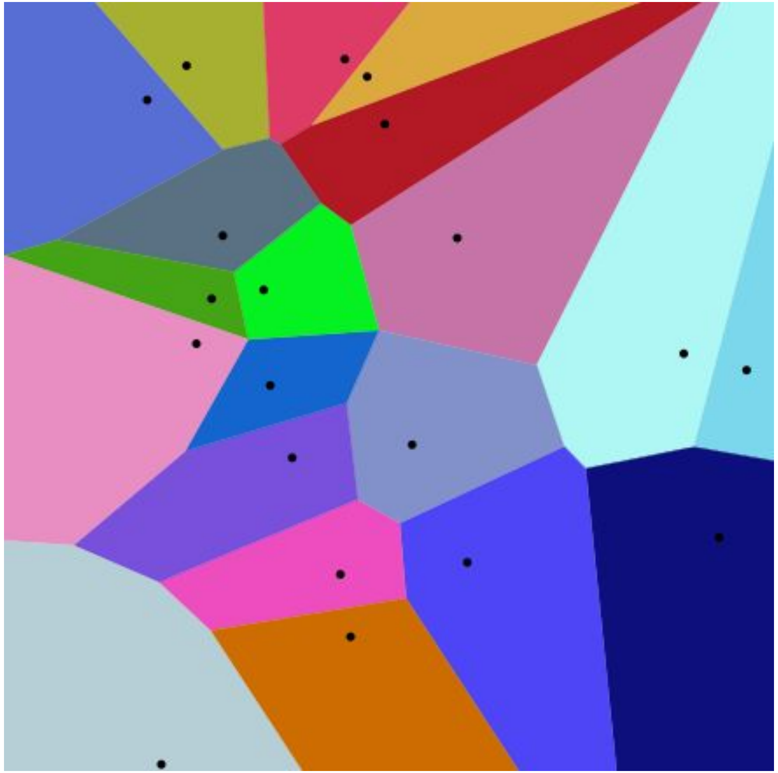


Image from wikipedia - Voronoi diagram

Method 2. Tessellation combined with GNN

VoroCNN

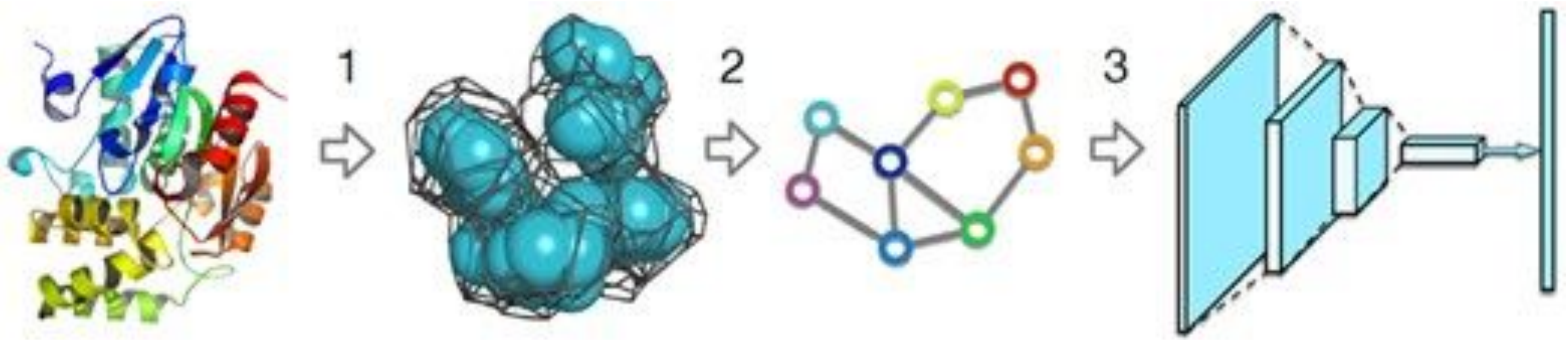


Image from Igashov et al, *Bioinformatics* **2021**

Method 3. Geometric learning

The challenge with 3D data is their non-euclidean distance

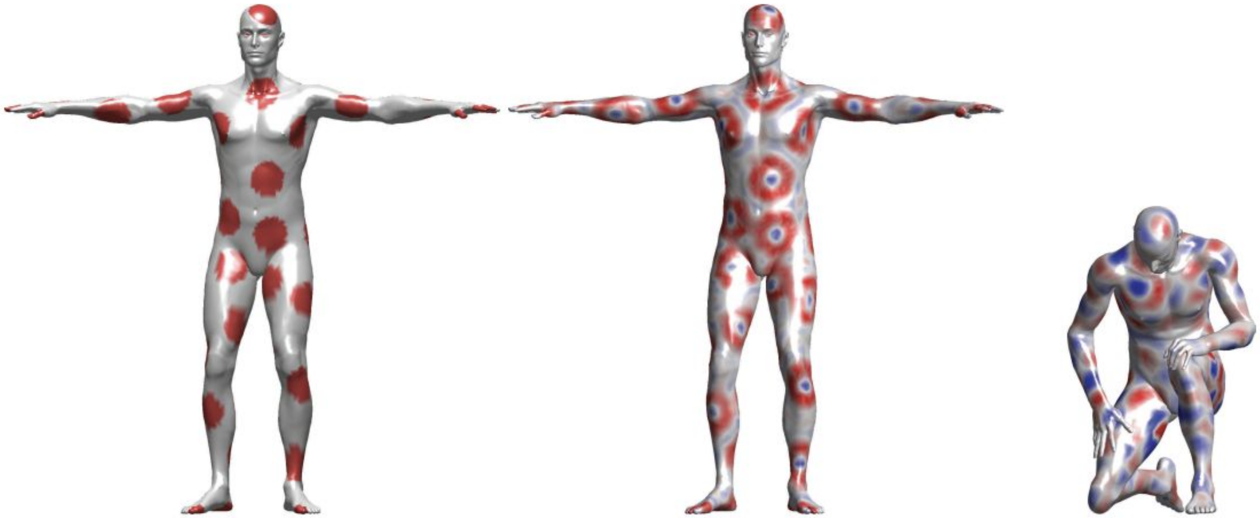
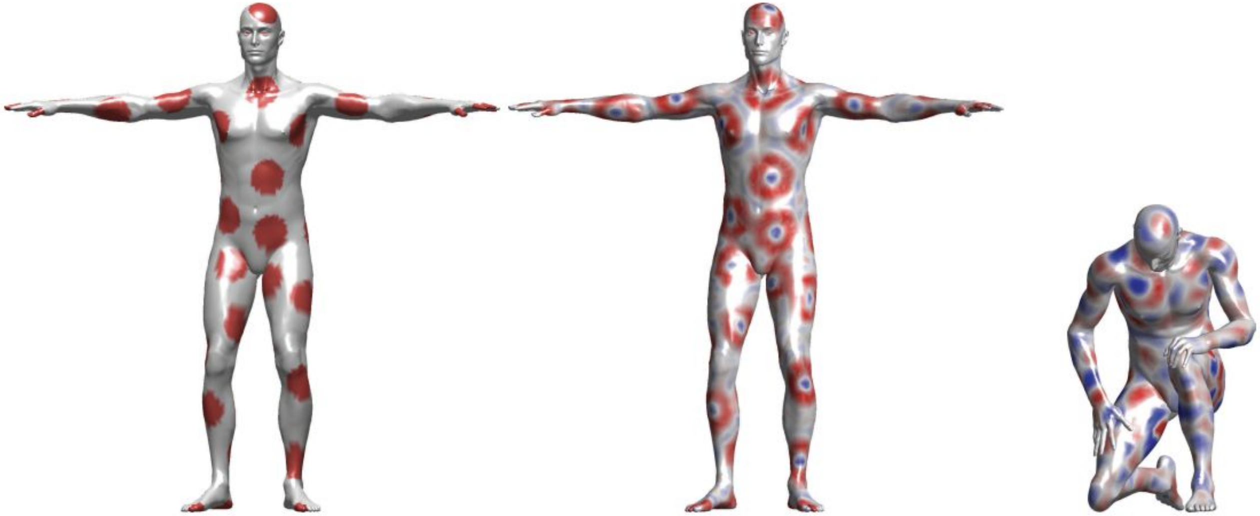
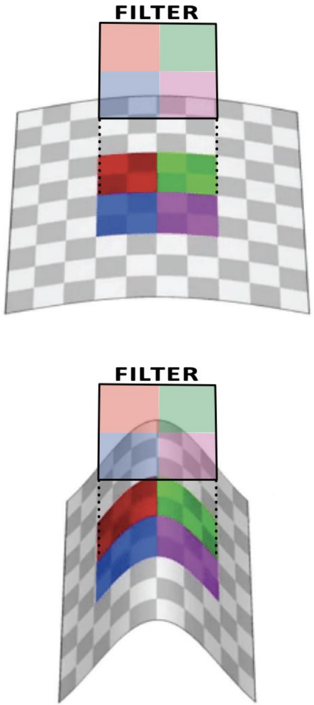


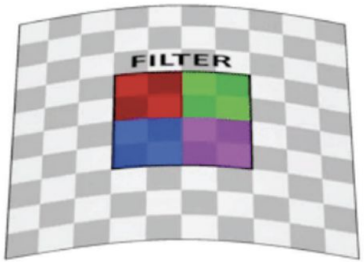
Image from Bronstein et al, *IEEE signal processing magazine* **2016**

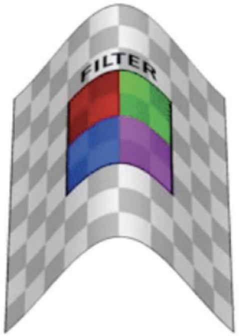
The conventional convolution layers don't work well when dealing with non-euclidean distances



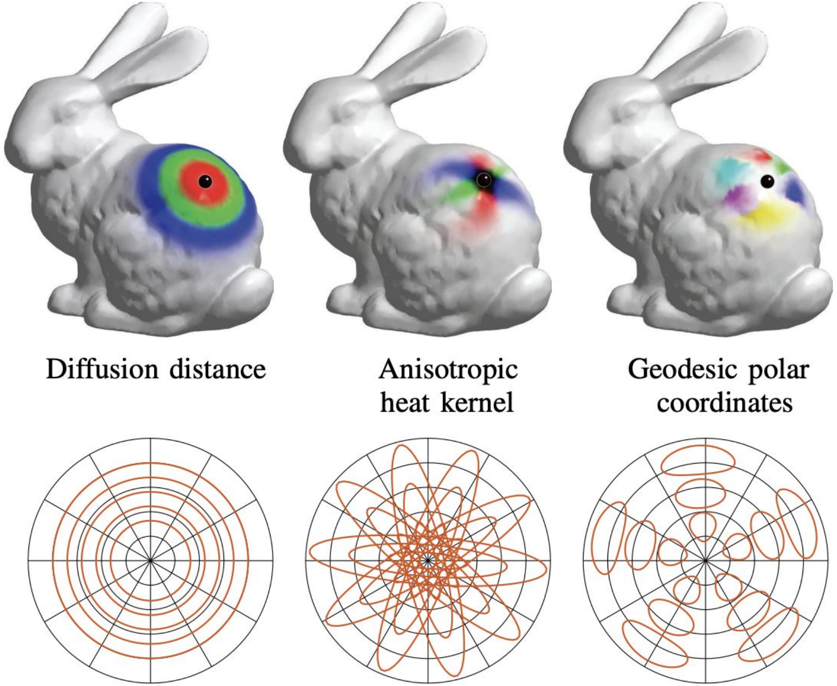


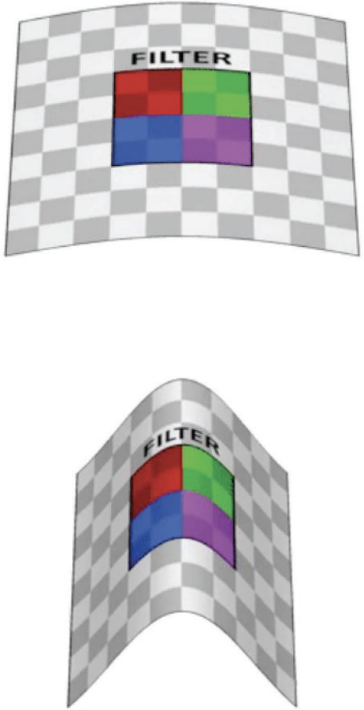
To solve this problem, geometric filters were developed



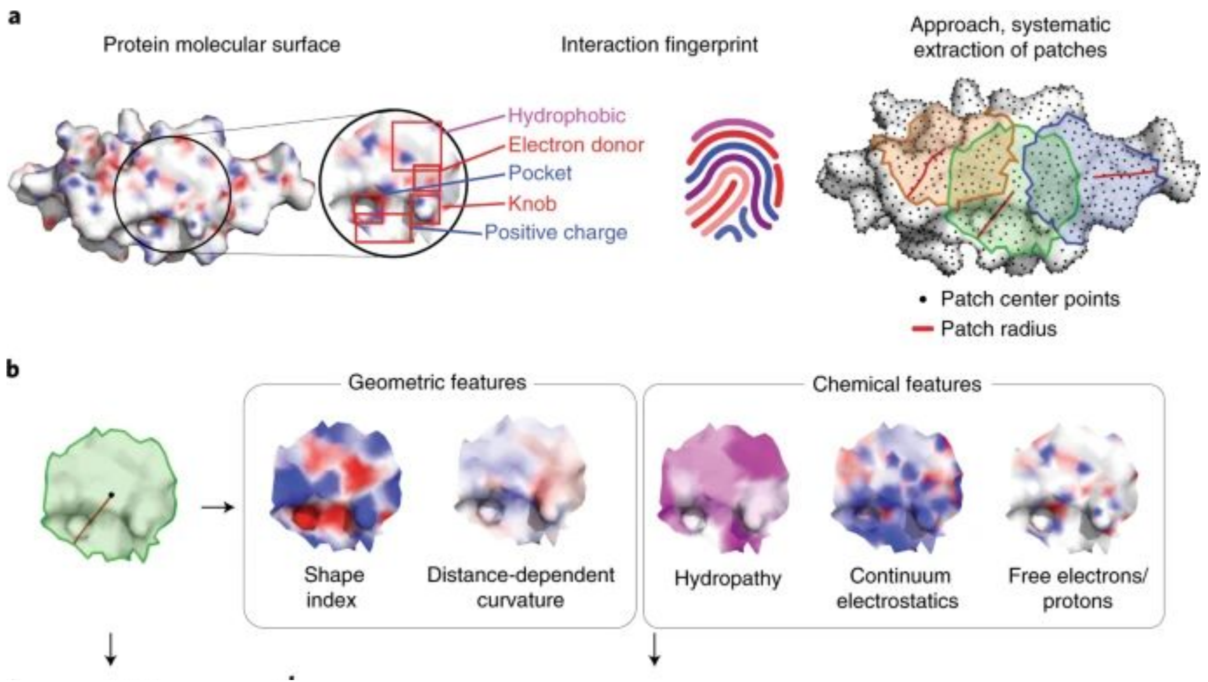


To solve this problem, geometric filters were developed





To solve this problem, geometric filters were developed



Method 4. Learning on point clouds

Method 4. Learning on point clouds

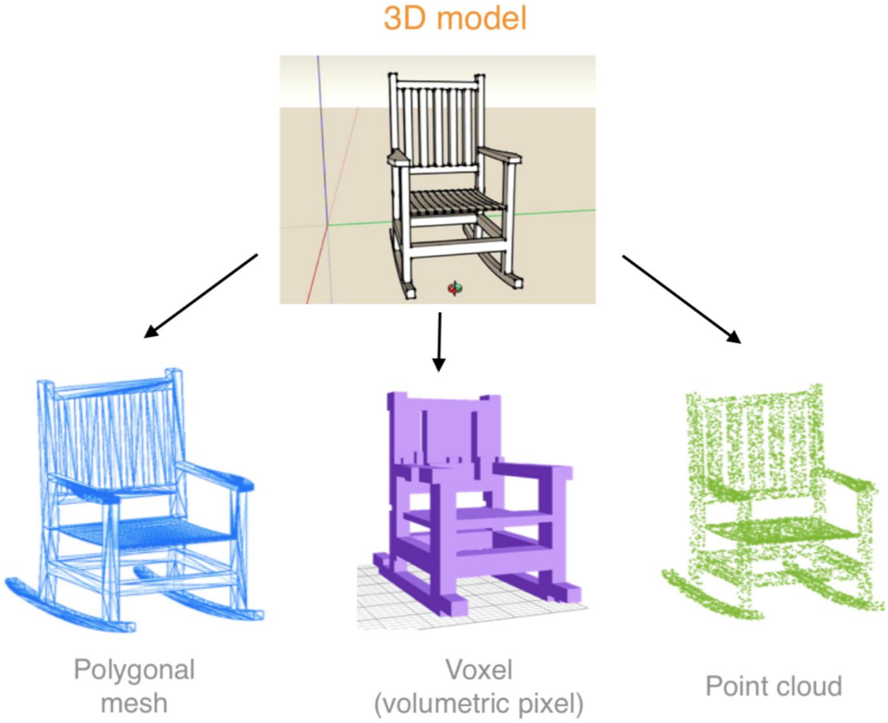
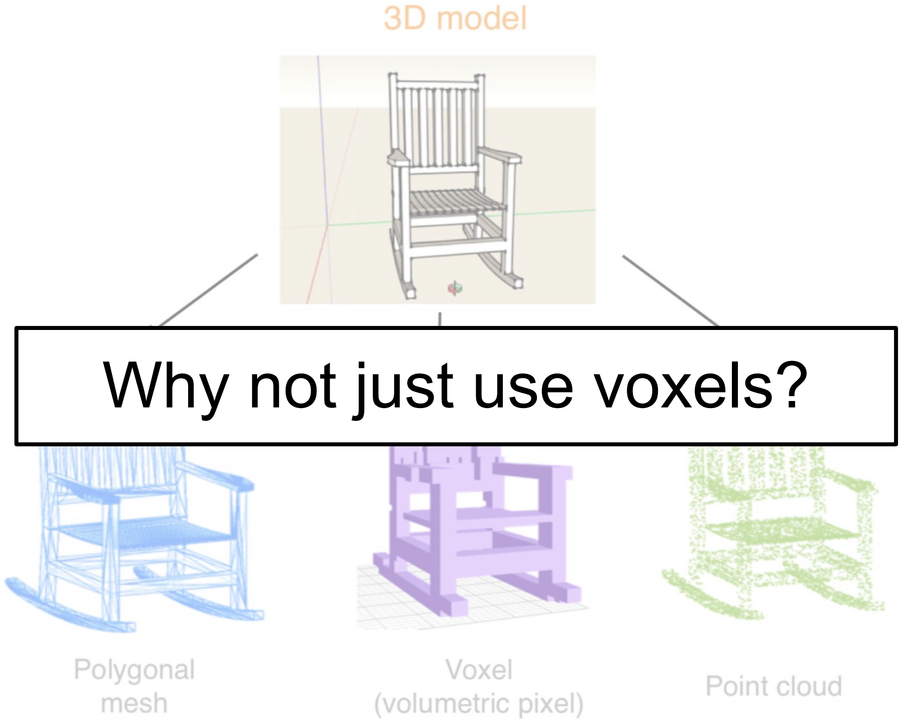


Image from TDS - Nikita Karaev: Implementing PointNet in google colab

Method 4. Learning on point clouds



Why not just use voxels?

Main challenges with point clouds

1. Unordered → invariant to permutation

Main challenges with point clouds

1. Unordered → invariant to permutation
2. Invariant to rigid transformation

Main challenges with point clouds

1. Unordered → invariant to permutation
2. Invariant to rigid transformation
3. Should capture interactions among points

PointNet++ is a leading algorithm that addresses all these challenges

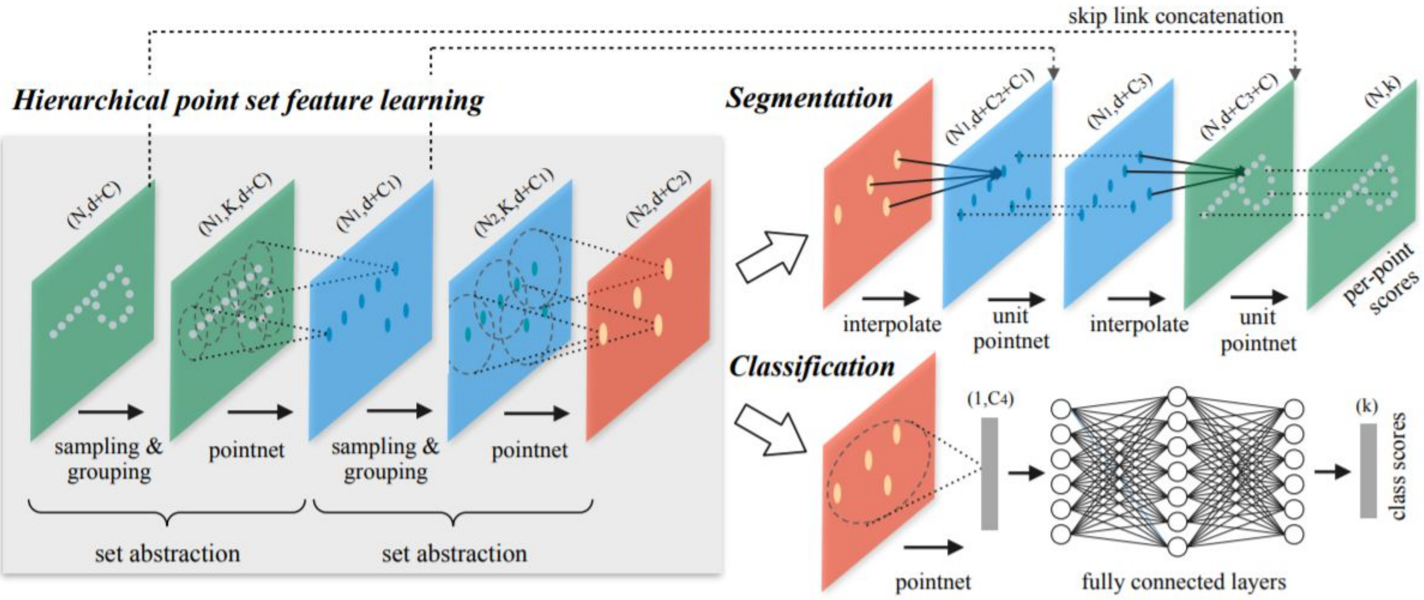
Use of symmetric functions

$x + y + z = y + z + x = z + y + x = \ldots$

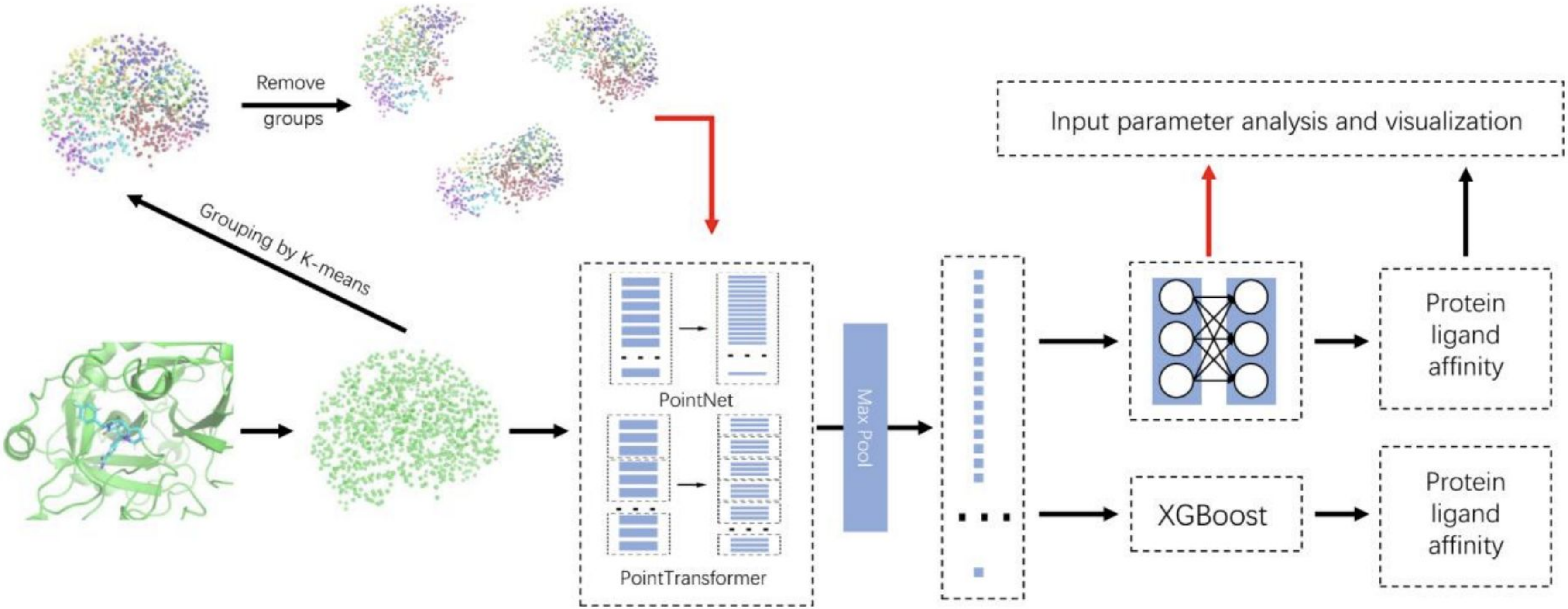
or

$\max(x, y, z)$

Overall architecture of PointNet++



PointNet on proteins for protein-ligand binding affinity prediction



Week 9, Lecture 1

Tips and tricks of deep learning

Data processing is one of the most important steps

1. Check to make sure that your data is balanced

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points
3. Replace missing points using median or 0

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points
3. Replace missing points using median or 0
4. Use data augmentation

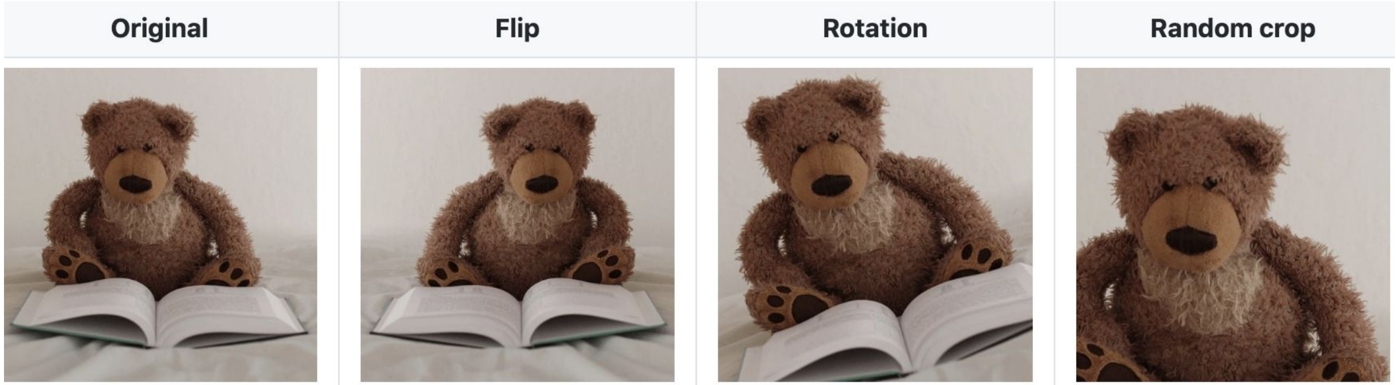


Image from Stanford CS230

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points
3. Replace missing points using median or 0
4. Use data augmentation
5. Think about your task and make sure your data matches your taks

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points
3. Replace missing points using median or 0
4. Use data augmentation
5. Think about your task and make sure your data matches your taks
6. Normalize features if your learning method is sensitive to it

Data processing is one of the most important steps

1. Check to make sure that your data is balanced
2. Remove outliers and noisy points
3. Replace missing points using median or 0
4. Use data augmentation
5. Think about your task and make sure your data matches your taks
6. Normalize features if your learning method is sensitive to it

Spend more time to get better data!!

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!
2. Check your initialization. Sometimes method other than random initialization help, such as *Xavier initialization*

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!
2. Check your initialization. Sometimes method other than random initialization help, such as *Xavier initialization*
3. Change learning rate. Use Adam as initial guess

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!
2. Check your initialization. Sometimes method other than random initialization help, such as *Xavier initialization*
3. Change learning rate. Use Adam as initial guess
4. Use *regularization* to avoid overfitting:
    a. Dropout
    b. Weight regularization

Tuning the network

L1 Regularization aka Lasso

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization aka Ridge

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

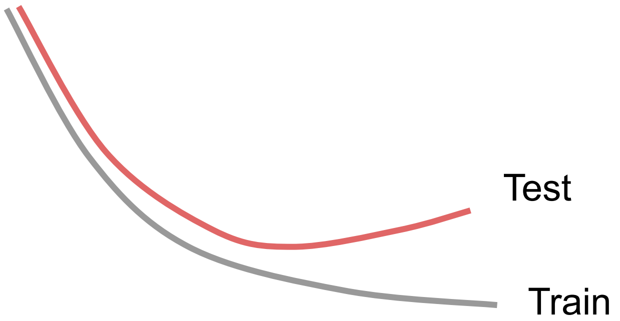
Image from medium - Dhaval Tanuk L1 vs L2 regularization

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!
2. Check your initialization. Sometimes method other than random initialization help, such as *Xavier initialization*
3. Change learning rate. Use Adam as initial guess
4. Use *regularization* to avoid overfitting:
   a. Dropout
   b. Weight regularization
5. Use early stopping

Tuning the network

1. Take advantage of pre-trained models. Use transfer learning!
2. Check your initialization. Sometimes method other than random initialization help, such as *Xavier initialization*
3. Change learning rate. Use Adam as initial guess
4. Use *regularization* to avoid overfitting:
   a. Dropout
   b. Weight regularization
5. Use early stopping



Good practices

1. Use mini-batch overfitting

Good practices

1. Use mini-batch overfitting
2. Check gradients

Automated hyperparameter tuning

Hyperparameter:

A parameter whose value is used to control the learning process

Automated hyperparameter tuning

Hyperparameter:

A parameter whose value is used to control the learning process
Unlearnable at times

Automated hyperparameter tuning

1. Choose parameters to tune

Automated hyperparameter tuning

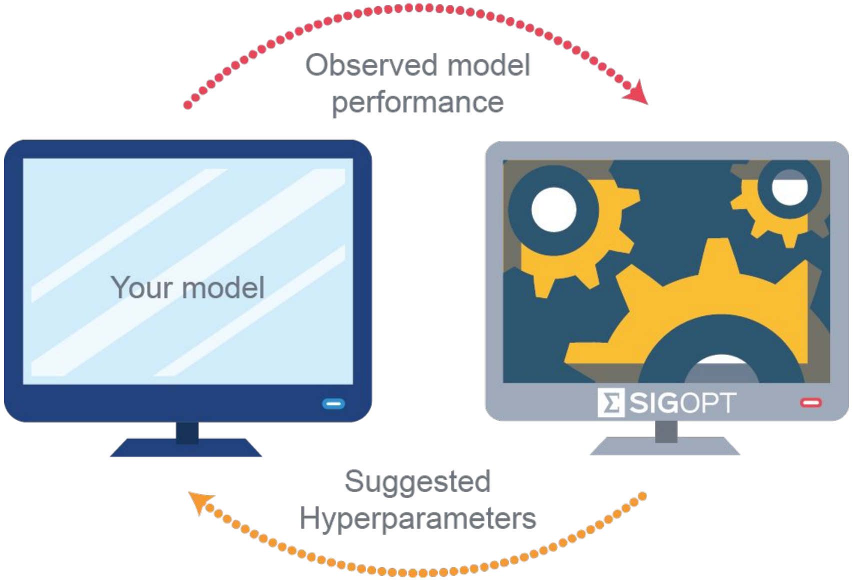
1. Choose parameters to tune
2. Set grids for each parameter

Automated hyperparameter tuning

1. Choose parameters to tune
2. Set grids for each parameter
3. Run an automated tuner

In-class activity

Hyperparameter tuning



Next lecture:
Responsible AI