Technische Universität München

Protein function prediction from 2D representation of 3D structure

Avinash Singh
Rajat Jain

Advisor-
Vladimir Golkov

Contents

What is the project about?	Phase 1 : Get it working!	Phase 2 : Add more features	Phase 3 : Add more data
Data	What we did	What is added	Include more classes
Distances b/w carbon atoms	How we did it	Results	Strict and Naive splits
Rotamers	Results		Results
Tools and language used			Conclusion

What is the project about?

- Motivation : Structural conformation decides what the protein can or cannot do
- Predict protein function using its structural information

Enzymes

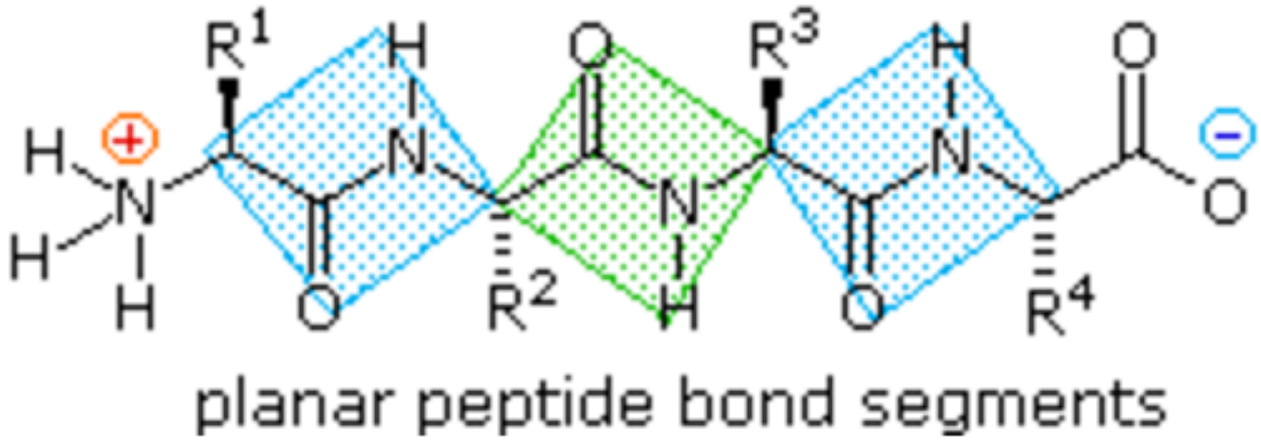
- **E.C.1.-.-.-** Oxidoreductases. [**6,069** PDB entries]
- **E.C.2.-.-.-** Transferases. [**11,995** PDB entries]
- **E.C.3.-.-.-** Hydrolases. [**15,538** PDB entries]
- **E.C.4.-.-.-** Lyases. [**2,554** PDB entries]
- **E.C.5.-.-.-** Isomerases. [**1,564** PDB entries]
- **E.C.6.-.-.-** Ligases. [**1,122** PDB entries]

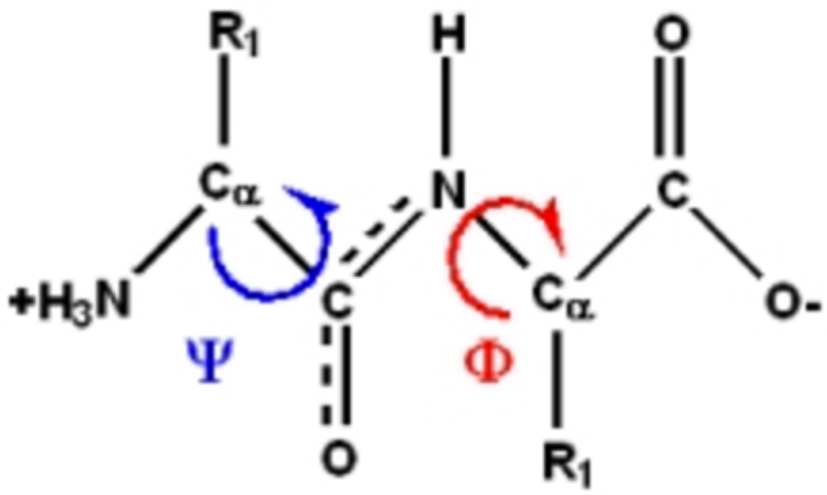
- Classify proteins into their enzyme classes on 3rd level

EC 3.4.21.- Serine endopeptidases. [**2,007** PDB entries]
EC 3.4.22.- Cysteine endopeptidases. [**751** PDB entries]
EC 3.4.23.- Aspartic endopeptidases. [**1,071** PDB entries]
EC 3.4.24.- Metalloendopeptidases. [**483** PDB entries]

Data

- Protein structure from PDB files eg. “1w3b.pdb”
- Distances between α-carbon and ß-carbon atoms
- Main chain and side chain angles
- Φ, Ψ : These two angles of rotational freedom allows polypeptides to fold up into unique conformations



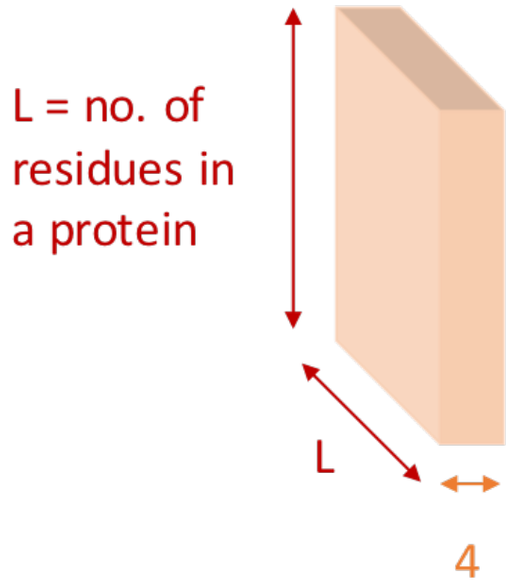


With support from

- Theano + Lasagne
- Python 2.7
- Convolutional Neural Networks

Phase 1

- Input : Distance matrix
- 4*L*L (L=no. of residues)
- 2 classes
  - EC 3.4.21._
  - EC 3.4.24._
- Training data ~ 800 proteins
- Validation data ~ 500 proteins

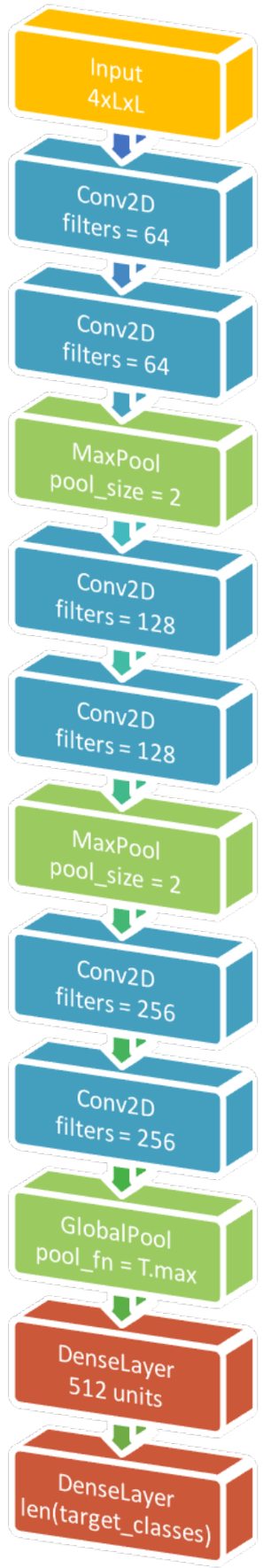


What we did

- Start small
- Store distance matrices on disk
- Learn about CNN/theano/lasagne
- Get it working
- Observe training and validation loss
- Iterate!

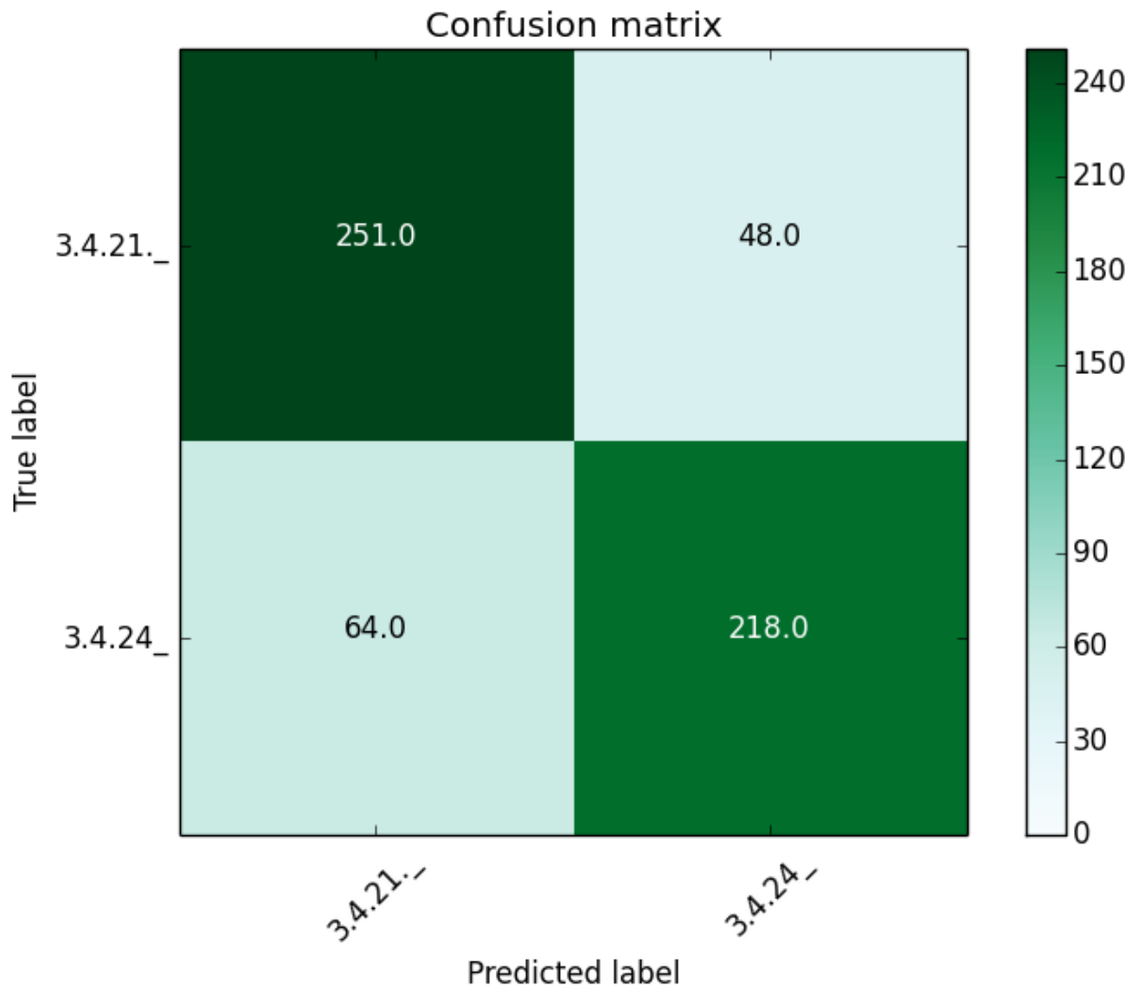
How we did it

- Network
  - filter size = 3
- Epoch ~ 40
- Loss function : Categorical cross entropy
- Learning rate = 1e-5
- Training time per epoch ~ 5 mins



Results

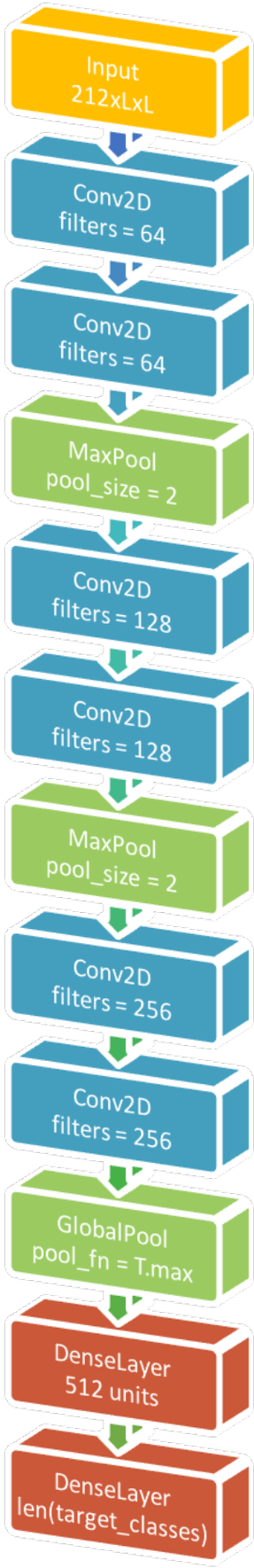
- Overall Accuracy = 80.72%
- Confusion matrix on test data
- Increasing number of layers didn't have any effect!



Phase 2 : Add more features

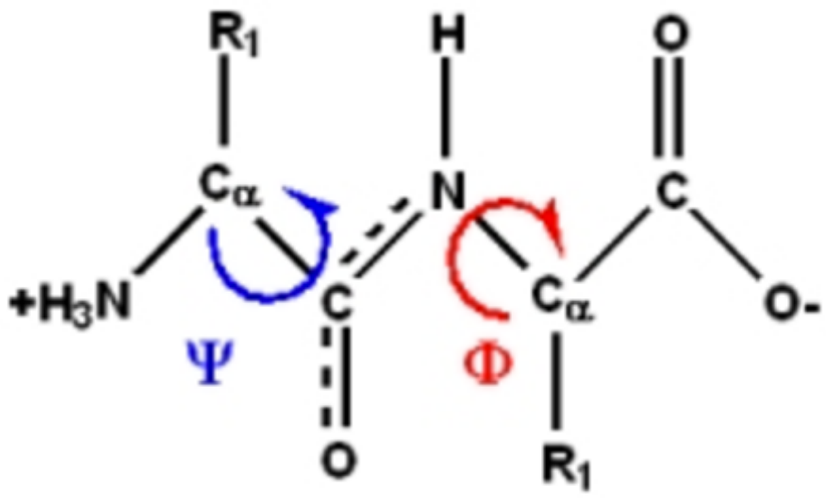
- Validation accuracy not improving much

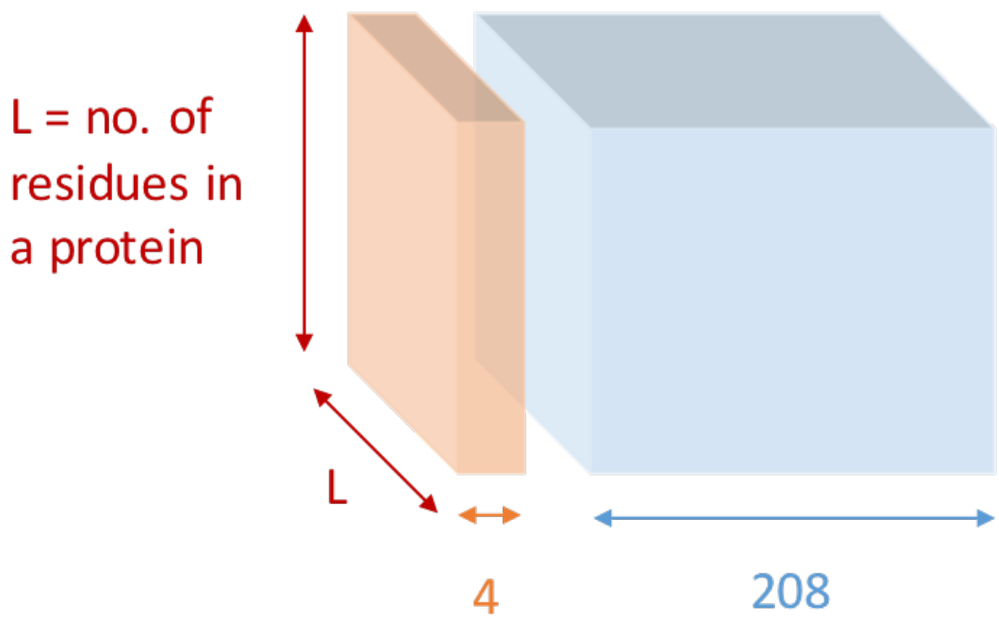
- Add new features



What we added

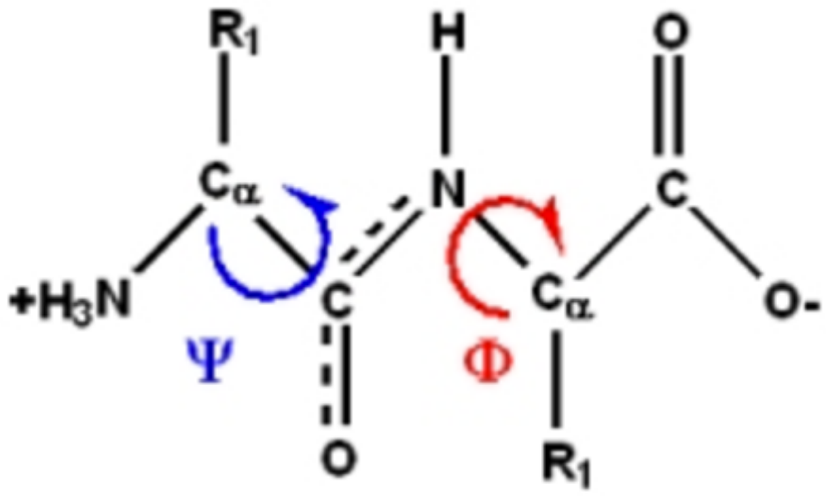
- Dihedral angles (Φ and Ψ)
- Side chain angles (chi1..5)
- Rotamers Input : 208*L*L
- Total input : 212*L*L
- Same 2 classes

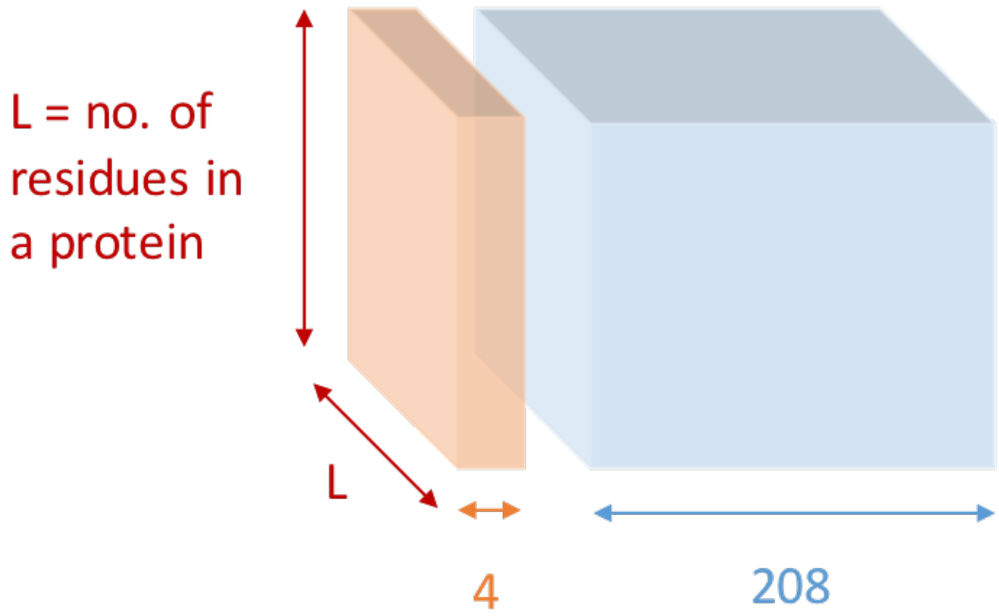




How 208?

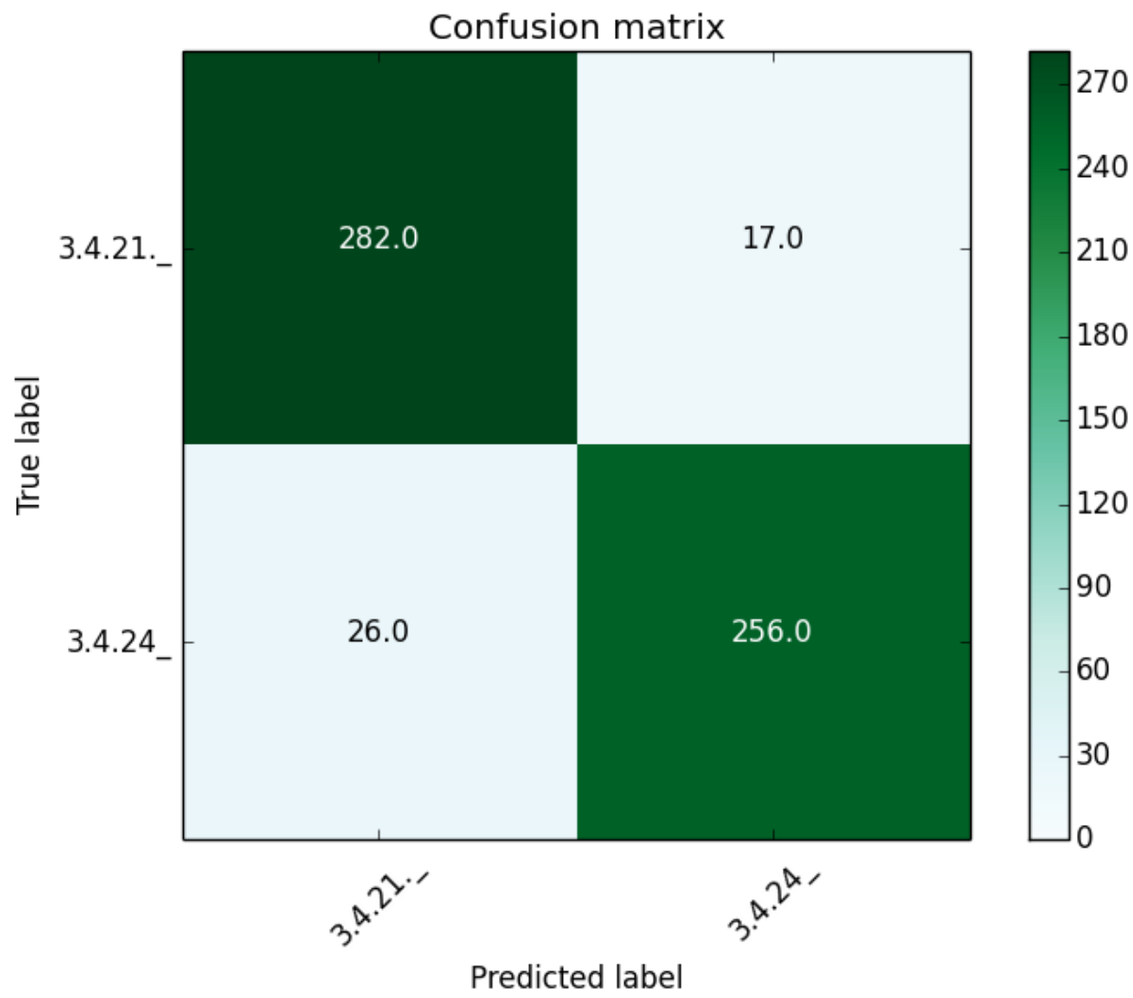
- Rotamers Input : 208*L*L
- Total side chain + main chain angles for one residue = 40 + 2
- Each with sine and cosine values
  - 0 in both if angle is not present
- One hot encoding = 20 amino acids
- So 42*2 + 20 = 104
- L*104 = stored on disk
- Reshape and tile on read





Results

- Accuracy improved = 92.50%
- Training times
  - 1 epoch ~ 12 mins
  - Total ~ 40 epochs
- Confusion matrix

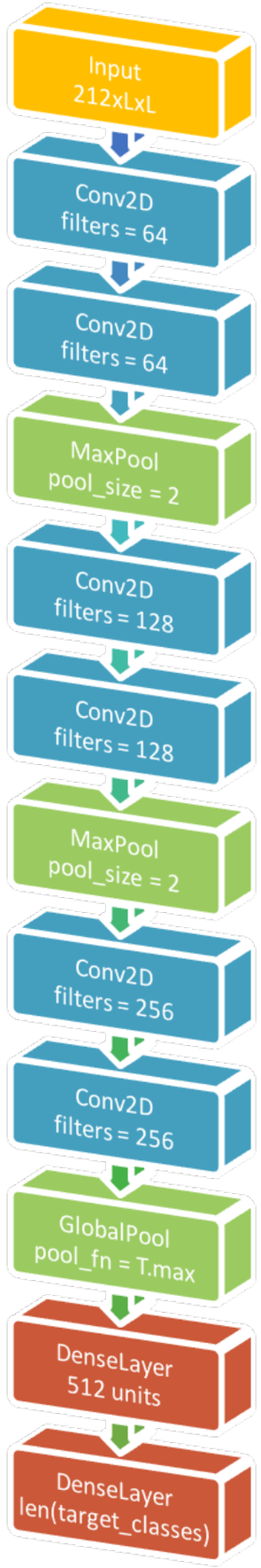


Phase 3 : Add more data!

- 4 classes:

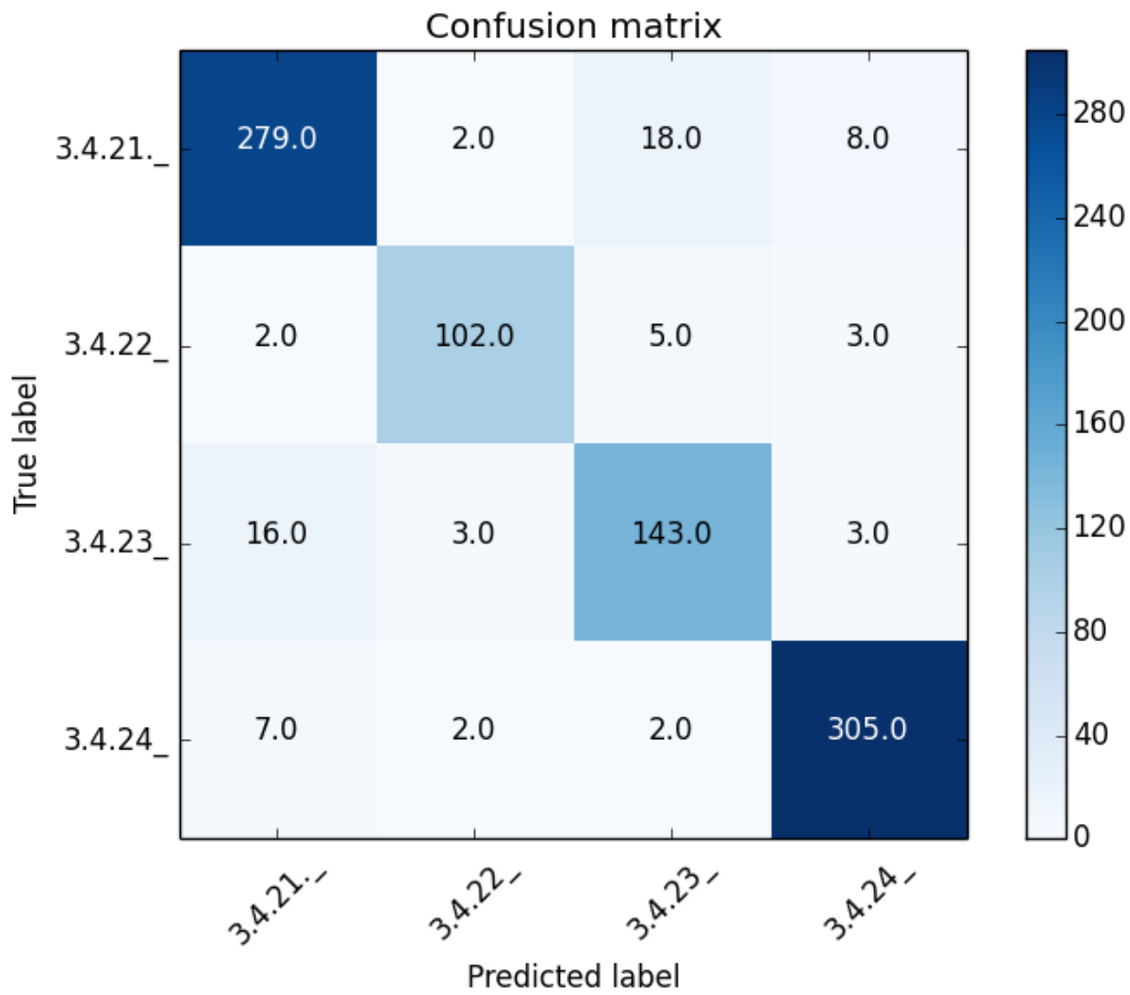
EC 3.4.21.- Serine endopeptidases. [**2,007** PDB entries]
EC 3.4.22.- Cysteine endopeptidases. [**751** PDB entries]
EC 3.4.23.- Aspartic endopeptidases. [**1,071** PDB entries]
EC 3.4.24.- Metalloendopeptidases. [**483** PDB entries]

- Training ~1950 proteins
- Validation ~ 900 proteins
- Split criteria :
  - Naive (Something from all)
  - Strict



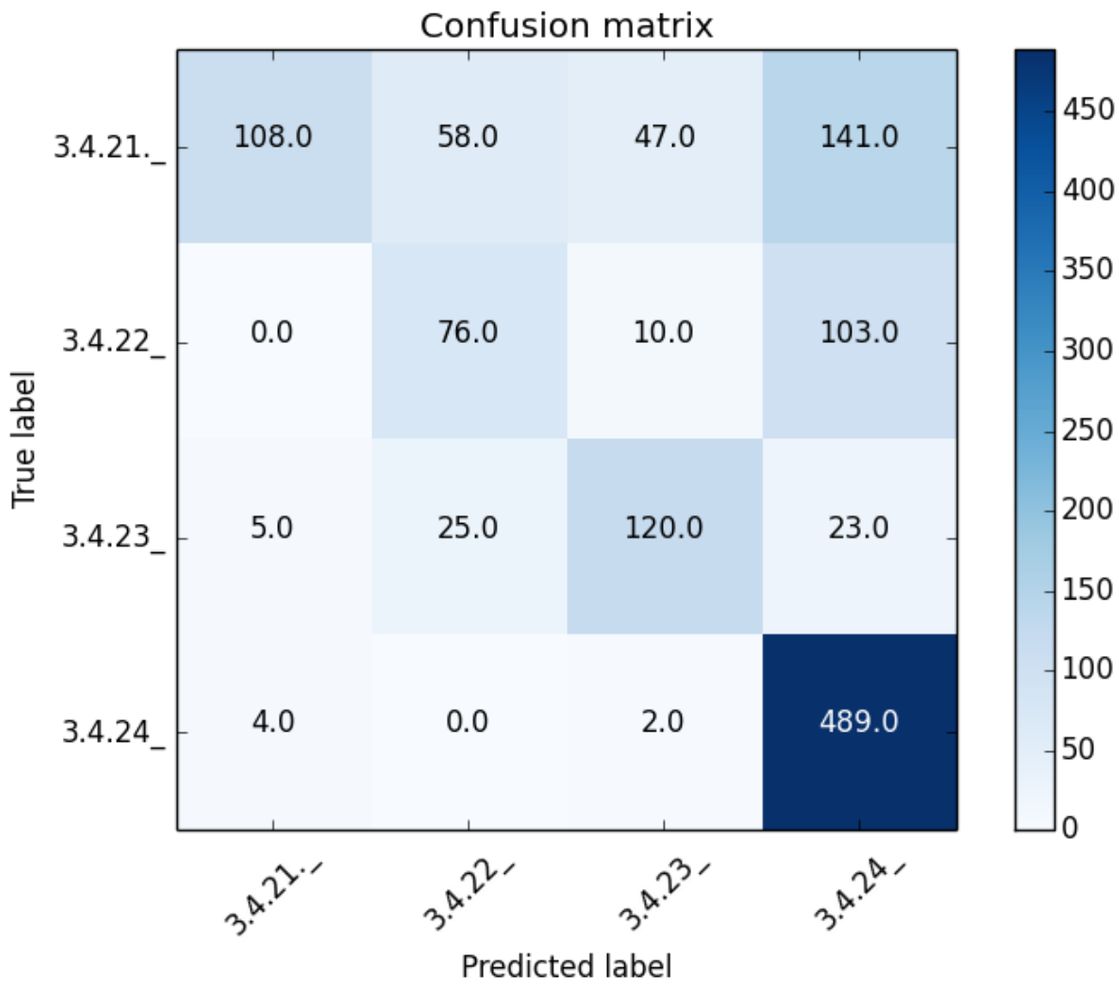
Results

- 4 class confusion matrix
- Naive split
- Training times
  - 1 epoch ~ 25 mins
  - Total ~ 50 epochs
- Overall accuracy = 92.11%



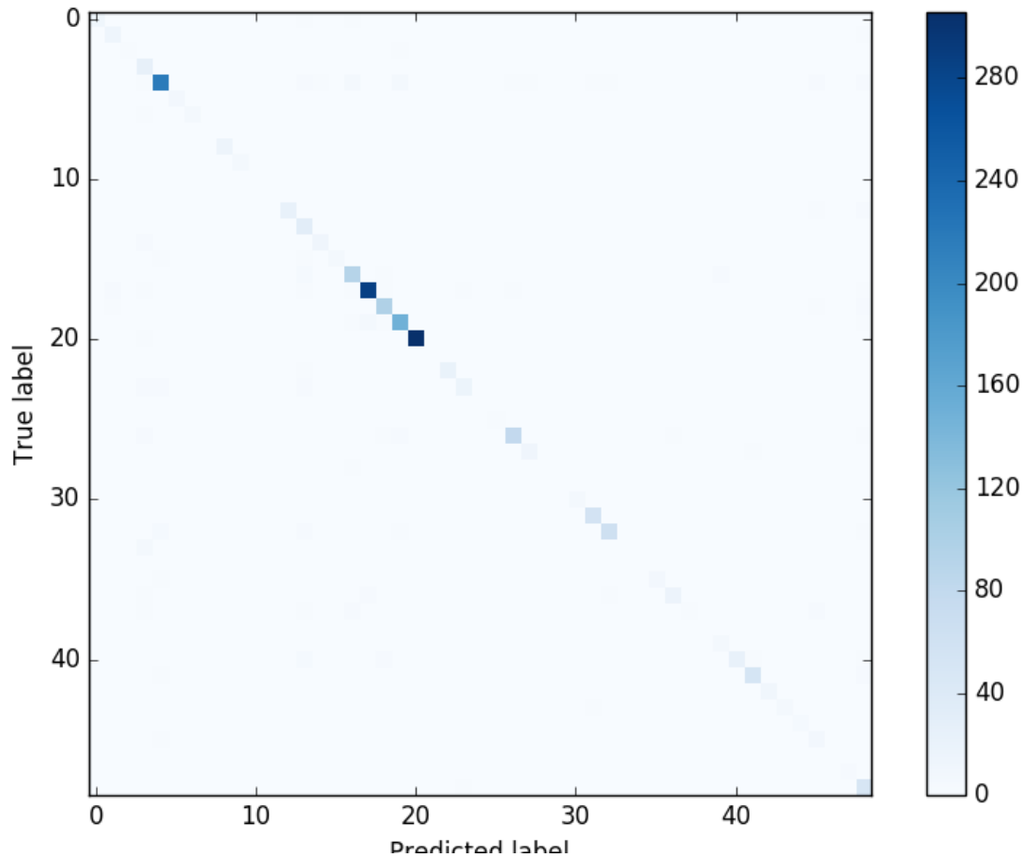
More Results

- 4 class confusion matrix
- Strict split
- Training times
  - 1 epoch ~ 25 mins
- Total ~ 50 epochs



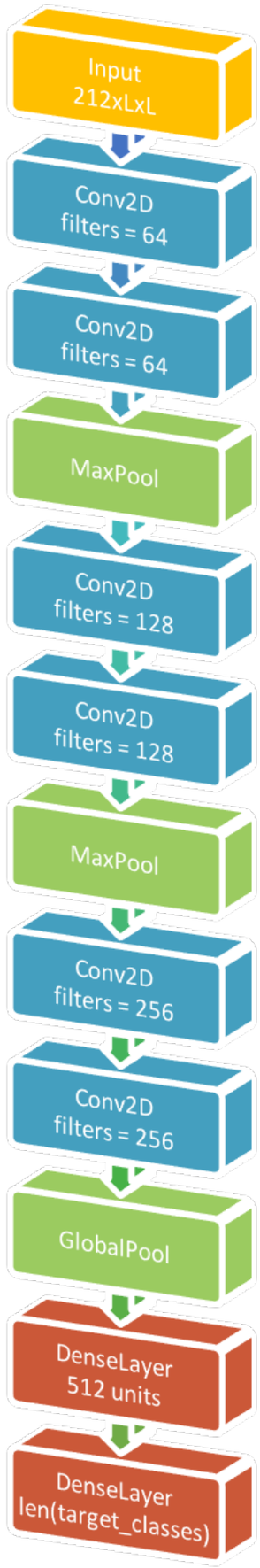
More Results

- 49 classes
  - EC 3.4.21.- to EC 3.4.24.-
  - EC 4.1.1.- to EC 6.6.1.-
- Confusion matrix
- Naive split
- Data
  - Train ~ 6000 proteins
  - Validation ~ 3000 proteins
- Training times
  - 1 epoch ~ 90 mins
  - Total ~ 50 epochs



Conclusion

- Predicting protein function from structural information in 2D representation is feasible
- Results jump around from epoch to epoch in strict split
  - smaller learning rate
  - learning rate schedule
- GPUs help in processing such data much faster (didn't time it but ran once on CPU and never tried it again)
- Deep learning using CNN gives good results



Thank you! :-)

Questions ?