1. Generating Field Features
   Amino acid encoding: We will use the following table to encode the amino acid types.

0	A	ALA					
1	C	CYS					
2	D	ASP					
3	E	GLU					
4	F	PHE					
5	G	GLY					
6	H	HIS					
7	I	ILE					
8	K	LYS					
9	L	LEU					
10	M	MET					
11	N	ASN					
12	P	PRO					
13	Q	GLN					
14	R	ARG					
15	S	SER					
16	T	THR					
17	V	VAL					
18	W	TRP					
19	Y	TYR					

   Atoms are grouped together as functional atoms and the encodings of the types are given here:

1	A	1	CB								
2	C	1	SG								
3	D	3	CG	OD1	OD2						
4	E	4	CG	CD	OE1	OE2					
5	F	6	CG	CD1	CD2	CE1	CE2	CZ			
6	G	1	CA								
7	H	5	CG	ND1	CD2	CE1	NE2				
8	I	3	CG1	CG2	CD1						
9	K	1	NZ								
10	L	3	CG	CD1	CD2						
11	M	3	CG	SD	CE						
12	N	3	CG	OD1	ND2						
13	P	2	CG	CD							
14	Q	4	CG	CD	OE1	NE2					
15	R	4	NE	CZ	NH1	NH2					
16	S	1	OG								
17	T	1	OG1								
18	V	2	CG1	CG2							
19	W	9	CG	CD1	CD2	NE1	CE2	CE3	CZ2	CZ3	CH2
20	Y	7	CG	CD1	CD2	CE1	CE2	CZ	OH		

21	O	OXT									

For each sample, after the true label of the residue, it is given as a list, starting with the functional atom type and then its coordinate, x, y, z. Here is the first sample in file 2.coor/pdb1a1x.txt. The protein is 1A1X with 108 residues and one chain. The sequence is GSAGEDVGAPPDHLWVHQEGIYRDEYQRTWVAVVEEETSFLRARVQQIQVPLGDAARPSHLLTSQ LPLMWQLYPEERYMDNNSRLWQIQHHLMVRGVQELLLKLLPDD and the first two are missing from the file. For the first one, the true label is 0 (A, “ALA” in the file).

Functional Atom Type	X	Y	Z
19	-1.52417945138	9.99200722163e-16	1.11022302463e-16
18	0.0	0.0	0.0
20	-2.15595479721	0.120719667363	1.05058920341
17	0.467884539026	1.42038095529	1.66533453694e-16
18	-3.54877373206	-0.12125876345	-1.3170889533
19	-3.9817096304	1.29806892164	-1.604343945
20	-5.16689225333	1.59825713777	-1.58202388227
17	-2.10596527666	-0.144062342264	-1.18676336944
19	-2.89646865139	3.92144155343	-3.61342434466
15	-4.85934775811	4.66605924354	1.1494357318
4	-2.35674611461	4.47839788738	-1.28730886486
18	-3.23505870358	3.58167710663	-2.171706308
2	-3.97243710019	5.25724163904	0.507059004224
4	-2.69215675116	4.50884003806	0.192918475903
15	-4.08367137766	6.44088521887	0.139641242747
20	-2.22282553208	3.15810746597	-4.30558113235
17	-3.00301253626	2.16520583775	-1.87372019982
19	-2.51179478672	7.06144003732	-5.11784501022
15	-4.34337308118	3.9814410992	-7.73299806357
4	-4.41168787174	5.77965616593	-6.15703814556
18	-3.09896121204	5.66133755048	-5.35718914598
4	-4.90010411389	4.44264331412	-6.70896043039
15	-5.858559497	3.86873633384	-6.13833265629
20	-2.80761166025	8.00699561979	-5.84726328378
17	-3.34586849035	5.10331056557	-4.02677579667
19	0.1681278407	8.7883342397	-4.52329297311
5	0.429921817545	7.46427441816	-1.90010016611
3	-0.688859175376	8.44978561072	-2.17096134681
18	-1.06388981857	8.45039392613	-3.68446114747
17	-1.69065525563	7.18073303994	-4.07398819533
18	1.94524994896	7.99090876437	-5.9498582971
19	3.24850725125	8.03137259745	-5.18130570798
20	3.31357176836	7.60188945117	-4.02377694563
17	0.775829249588	7.76770939383	-5.12003405736
5	6.66705615984	8.75509169263	-6.32190205676

18	5.60878379004	8.66865803577	-5.22650449683
17	4.27737560306	8.58421752442	-5.8218792021
4	7.14892724092	8.23588096547	-2.77911749063
4	8.17613437102	8.61896176039	-1.74575568737
15	8.5731637362	4.95184986191	-0.668335617383
19	7.47206635656	4.83044753005	6.33518721291
3	8.27366488152	6.0921537202	4.32243802473
18	8.40794565786	5.94092746056	5.85700287885
5	6.82075000461	6.3883400397	3.94393964353
20	7.49906120941	3.72609206228	5.79819626444
19	4.501399093	4.03491678699	7.13732363899
18	5.79407693236	4.08503434507	7.93412300771
20	3.86101583687	2.9855683218	7.03360245559
17	6.71752265834	5.08925861101	7.40018797903
19	2.90861748333	6.51694872081	4.8915940864
6	0.307505129778	6.07636217029	8.35644675509
4	1.60000254419	4.91313300224	6.43755271519
18	2.93586558719	5.24514487143	5.73462523027
6	0.969479019459	6.05032947246	7.1783882387
20	3.4768025479	7.54625074541	5.26680479327
17	4.09385193742	5.18829187761	6.61822035801
6	0.234489449177	8.09850411564	7.51625650089
8	0.906819074848	7.33311956656	6.67816798141
8	-0.138806490147	7.3603080721	8.54366817251
19	0.831027474393	7.30820422126	2.15463254341
4	3.2796249782	7.32513354264	1.64375715507
18	2.22081198965	7.467748259	2.74075173024
6	3.21883751652	8.39028547097	0.589803999528
20	0.595084784262	6.46514866019	1.2929937746
17	2.34696970673	6.38102554595	3.69833356766
8	3.78778233415	8.23251337265	-0.655002716185
5	-2.28779950868	7.92587226448	5.4000888192
5	-2.33494093939	5.5027496222	5.02614985898
4	-2.25027636798	7.05215208908	3.07542130125
18	-1.46442795695	8.0401945387	2.19523473431
3	-1.80654515288	6.81874219028	4.52315139622
17	-0.0928466788292	8.12221572148	2.64274710631
19	5.88909330321	8.74564478516	8.62549281826
20	6.14551717748	7.62060452025	8.17544704387

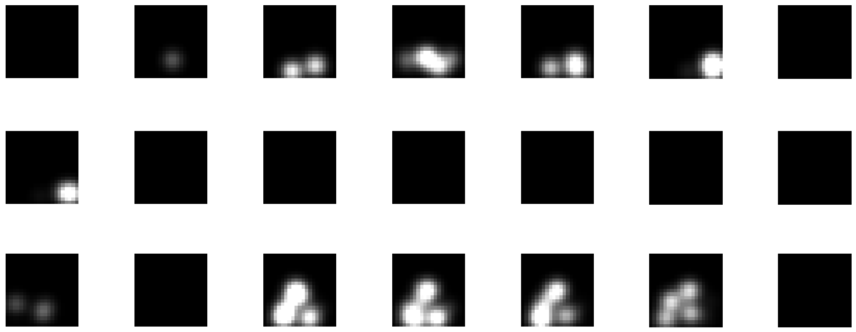
In this case, there are 74 rows. First we need to read a sample into a two-dimensional matrix of size nrow x 4 of type double. The field feature map will be a four-dimensional tensor [X, Y, Z, Channel].
To compute the four-dimensional tensor from a two dimensional matrix, we need to define some hyperparameters.

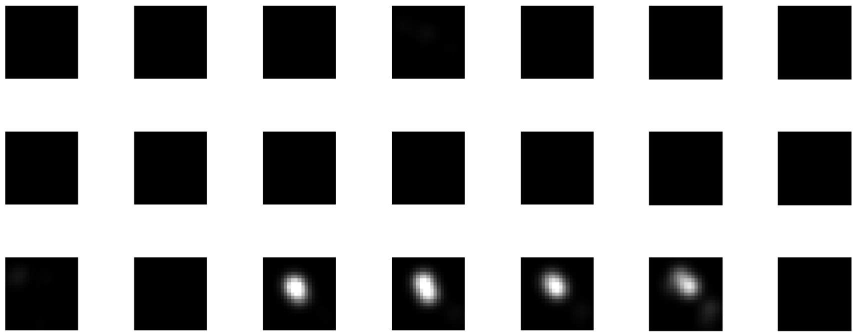
Name of the Hyperparameter	Range	Default Value	Comments
Window_width	0.1 - 10	2	It controls the size of the Gaussian window for each data point
Grid_cell_size	0.1 – 2	1	It controls the size of the generated volume
Grid_X_Size	10 - 100	21	
Grid_Y_Size	10 – 100	21	
Grid_Z_Size	10 - 100	21	
Coord_X_min	-20 - 0	-10	
Coord_Y_min	-20 - 0	-10	
Coord_Z_min	-20 - 0	-10	

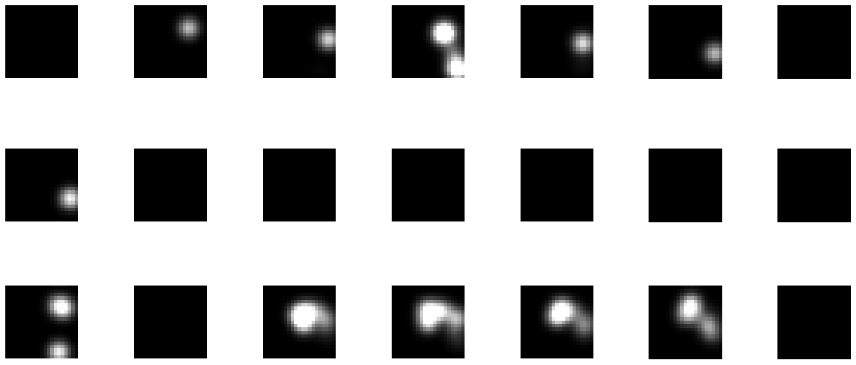
Let us name the two-dimensional matrix as “double coord_list[nrow][4];” and the four dimensional tensor as “double field_maps[Grid_X_Size][Grid_Y_Size][Grid_Z_Size][21]”. The pseudo code in C notation would be:

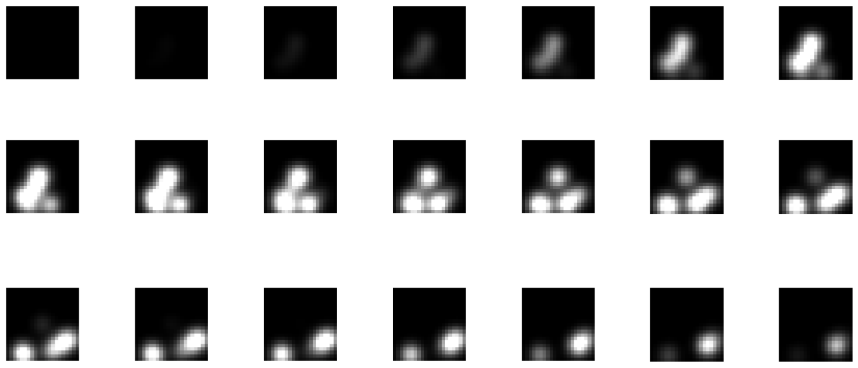
```
Compute_Field_Map_From_Sample:

Initialize all the elements in field_maps to 0

For ii=0 to Grid_X_Size-1
  Xx = (ii*Grid_cell_size)+ Coord_X_min
  For jj=0 to Grid_Y_Size-1
      Yy = (jj*Grid_cell_size)+ Coord_Y_min
      For kk=0 to Grid_Z_Size-1,
        Zz = (kk*Grid_cell_size)+ Coord_Z_min
        For mm=0 to (nrow-1),
           Channel = (int)(coord_list[mm[0]+0.5); // This is the functional atom type
           DistanceSq = $(Xx- coord\_list[mm[1]])^2+(Yy- coord\_list[mm[2]])^2+(Zz- coord\_list[mm[3]])^2$;
           Contribution = $\exp(-DistanceSq/(2* Window\_width^2))$;
           field_maps[ii][jj][kk][Channel] += Contribution;
        end
    end
end
```

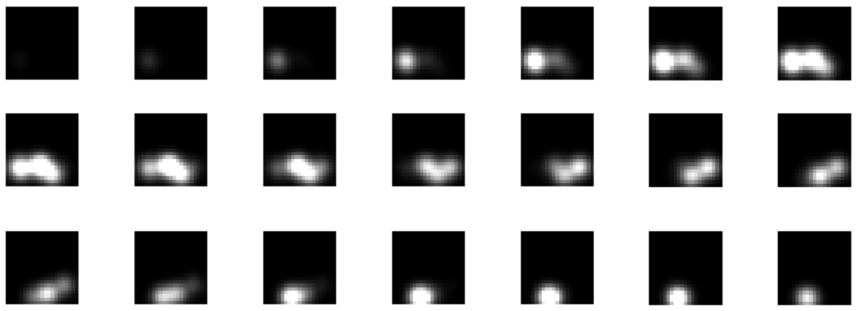
For the given example, here are some cross sections:


X=10 in the Y-Z plane for all the 21 atom types


Y=10 in the X-Z plane for all the 21 atom types


Z=10 in the X-Y plane for all the 21 atom types


The 3D stack (sliced along X) for atom type 18 (12 of them in the sample)


The 3D stack (sliced along X) for atom type 4 (9 of them in the sample)

For each sample, we have a four-dimensional tensor. For nsamples, we will have a five-dimensional tensor.

2. Designing and Training the Convolutional Network
The convolution neural network is very similar to a standard neural network except the convolution needs to be done using tf.nn.conv3d. It may be the best to organize the data as a 5-dimensional tensor in the following shape [batch, in_depth, in_height, in_width, in_channels]; for details, see https://www.tensorflow.org/api_docs/python/tf/nn/conv3d . Here the batch is the batch size. When we train the CNN, we do the training batch by batch. We can use a batch of 100 or any other value that works well. The depth is X, height is Y, and width is Z, and the channel is the number of functional atoms (21).
To design the CNN, we need to specify how many filters and their sizes. Clearly we need to experiment with the choices. Since it is not clear which functional atoms interact with any other ones, we need to use a filter of size 21 in the channel dimension. Initially we can use 21 5x5x5x21 filters in the first layer. We will use the default data format "NDHWC", which is the same as the one we have. At the end of the first layer, we will do maximum pooling. After the first layer, we can use another layer of 21 3x3x3x3 filters, and so on. After the number of convolution layers, we will have a fully connected layer and a soft-max layer for classification.

3. Experimental Results