

**Final Project**

STAT – 517

*Abhishek Kumar Thakur*

Materials Science & Engineering

**University of Idaho**

## **Abstract**

Microstructure plays a very important role in determining the properties (like strength, ductility, formability etc.) of a material. The final microstructure of a material depends upon a number of variables such as elasticity constant of matrix and precipitate, direction of strain applied on the material, temperature and so on. The experimental determination of the microstructure of a material is a very tedious task and requires a lot of effort. In this work we use machine learning methods to train a convolutional neural network over a range of different microstructures and using the trained model to predict the microstructure of a material under different circumstances.

## **Introduction**

The study of microstructure of a material is very helpful in designing a material for a given purpose. For different purposes different materials are required. For example, we need a material which is having high strength and high toughness for defense purposes. The final properties of a material depends upon the microstructure of that material which further depends upon a lot of variables like elasticity constants, applied strain and many more. This makes study of the properties of a material a very tedious task.

Machine learning can be a useful tool in such a case where there are a lot of parameters involved in determining the final output. Through convolutional neural network we can make the machine learn the characteristic properties of the material under different processing conditions. Further the developed model can be used to predict the microstructure of the material at any arbitrary set of parameters (processing conditions). This methodology can drastically reduce the efforts of experiments as the developed model is used for analyzing the properties of the material.

## **Methodology**

In this work, we have used a convolutional neural network to study the abstract features from the microstructure of a material. We have used eight features to define a particular microstructure keeping all other parameters constant. These eight parameters include the elasticity constants of the matrix in the three directions ( $c_{110}$ ,  $c_{120}$  and  $c_{440}$ ), the difference between the elasticity constants of the matrix and the elasticity constants of the precipitate in the three directions ( $c_{111}$ ,

c121 and c441) and externally applied strain in the x-direction (xs) and in the y-direction (ys). The network used to store the convolved feature of the input image is 5 layers deep. We have used leaky ReLU to model the negative parts of the input image. The image of the microstructure is represented by a  $512 \times 512$  size pixel data as shown in Figure 1. The pixel data is normalized between -1 and 1 where -1 represent black and 1 represent white.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.94521	0.94482	0.94345	0.94062	0.93494	0.92255	0.89305	0.81464	0.6249	0.25632	-0.23545	-0.61991	-0.81814	-0.89578	-0.92235	-0.9303	-0.93127	-0.93026
1	0.92889	0.93104	0.9281	0.91902	0.8972	0.84273	0.71303	0.44516	0.01341	-0.43482	-0.72441	-0.85864	-0.90896	-0.92654	-0.93147	-0.93191	-0.93081	-0.92877
2	0.84741	0.86251	0.85127	0.80772	0.70715	0.50777	0.16643	-0.26283	-0.60714	-0.80133	-0.88556	-0.91722	-0.92875	-0.93183	-0.93227	-0.93135	-0.92967	-0.9276
3	0.46445	0.51316	0.47922	0.35371	0.11155	-0.21881	-0.52792	-0.74113	-0.85391	-0.9021	-0.92183	-0.92923	-0.93153	-0.93221	-0.93169	-0.93062	-0.92903	-0.92686
4	-0.33994	-0.28963	-0.32634	-0.4371	-0.58849	-0.73312	-0.83342	-0.8867	-0.91218	-0.92408	-0.92885	-0.93092	-0.9319	-0.93183	-0.93147	-0.93043	-0.92893	-0.9267
5	-0.78989	-0.77549	-0.78734	-0.81922	-0.8564	-0.88656	-0.90682	-0.91928	-0.92572	-0.92876	-0.93073	-0.93173	-0.9321	-0.93223	-0.93175	-0.93088	-0.92937	-0.92717
6	-0.90007	-0.89866	-0.90117	-0.9069	-0.91408	-0.92091	-0.9256	-0.92823	-0.93007	-0.93158	-0.93226	-0.93289	-0.93313	-0.93309	-0.93259	-0.93184	-0.93015	-0.92808
7	-0.92601	-0.92653	-0.92763	-0.92914	-0.93038	-0.93118	-0.93199	-0.93299	-0.93355	-0.93388	-0.93431	-0.93451	-0.93448	-0.93434	-0.93376	-0.93283	-0.93132	-0.92879
8	-0.93365	-0.93465	-0.93531	-0.93571	-0.93612	-0.9365	-0.93658	-0.93633	-0.93633	-0.93638	-0.93629	-0.93614	-0.93602	-0.93548	-0.93495	-0.93374	-0.93233	-0.9294
9	-0.93857	-0.9395	-0.94014	-0.94041	-0.94026	-0.93976	-0.93939	-0.93912	-0.93878	-0.93829	-0.93807	-0.93761	-0.93719	-0.93651	-0.93577	-0.93438	-0.93291	-0.92978
10	-0.94106	-0.942	-0.94247	-0.94255	-0.94246	-0.94224	-0.94175	-0.94103	-0.94045	-0.93993	-0.93929	-0.93866	-0.938	-0.9371	-0.93611	-0.93467	-0.93285	-0.92992
11	-0.9433	-0.9441	-0.94452	-0.94465	-0.94429	-0.9437	-0.94301	-0.94245	-0.94165	-0.94085	-0.94006	-0.93928	-0.93826	-0.93734	-0.93595	-0.93451	-0.9323	-0.92965
12	-0.94441	-0.94513	-0.94541	-0.94535	-0.94509	-0.94469	-0.944	-0.94311	-0.94222	-0.94138	-0.94035	-0.93936	-0.93823	-0.937	-0.93549	-0.93382	-0.9315	-0.92883
13	-0.94536	-0.94587	-0.94613	-0.9461	-0.94569	-0.94496	-0.94419	-0.94338	-0.94243	-0.9413	-0.94026	-0.93905	-0.93778	-0.93634	-0.93473	-0.93278	-0.93055	-0.92766
14	-0.94556	-0.94603	-0.94611	-0.94588	-0.9455	-0.94499	-0.94419	-0.94316	-0.94209	-0.94103	-0.93975	-0.93845	-0.93704	-0.93546	-0.93337	-0.93169	-0.92933	-0.92663
15	-0.94559	-0.94581	-0.94589	-0.94575	-0.94525	-0.94445	-0.94356	-0.94267	-0.94157	-0.94029	-0.93903	-0.93763	-0.93608	-0.93445	-0.93259	-0.93052	-0.92821	-0.9256
16	-0.94506	-0.94531	-0.94518	-0.94479	-0.9443	-0.94374	-0.94291	-0.9418	-0.94064	-0.93947	-0.93808	-0.93662	-0.93507	-0.93333	-0.93148	-0.92942	-0.92716	-0.92475

Figure 1: Part of the microstructure data (original data is  $512 \times 512$  size)

We have used 2048 microstructures in the whole process out of which 1648 is used for training the network while 400 is used for testing purpose. Mean squared error loss is used to compute the loss function which indicates how well the machine is learning through the given input images. The training of the convolutional neural network is done by feeding the images of the microstructures into the network in mini-batches of size 64. The network is trained for 8000 epochs during which the weights and losses are updated after every epoch.

## Supervised learning

We have done a thorough analysis of the phase field and machine generated images. This exercise is being done in order to monitor the efficiency of machine generated images. Basically we have used three methods for analyzing and predicting the efficiency of machine generated images.

### 1. Particle analysis (using ImageJ software)

In this method we have used ImageJ software for particle analysis. We essentially calculate the area fraction of the matrix and precipitate for both phase field ( $PF$ ) and machine generated ( $M$ ) image. The error is calculated by using following equation:

$$\%Error_{Matrix} = \frac{|\%Matrix_{PF} - \%Matrix_M|}{\%Matrix_{PF}} \times 100$$

$$\%Error_{Precipitate} = \frac{|\%Precipitate_{PF} - \%Precipitate_M|}{\%Precipitate_{PF}} \times 100$$

We have also defined an efficiency term ( $\eta$ ) in order to monitor the predicting capacity of machine after learning through the neural network model. This is shown as follows:

$$\eta = 1 - \frac{|\%Matrix_{PF} - \%Matrix_M|}{\%Matrix_{PF}}$$

$$\eta = 1 - \frac{|\%Precipitate_{PF} - \%Precipitate_M|}{\%Precipitate_{PF}}$$

## 2. Using the random lines method.

In this method, we use ImageJ software. We draw 5 random lines (can draw more or less random lines) over the image and calculate the number of intersection for each line with the black and white region of the image. Then we get the average of the length of those 5 random lines ( $l_{avg}$ ) and the average of the number of intersections for each random line ( $I_{avg}$ ). Then we report the number of intersections per unit length of random line ( $\rho$ ) as follows:

$$\rho = \frac{I_{avg}}{l_{avg}}$$

Further the %Error and efficiency ( $\eta$ ) is calculated using the following equations:

$$\%Error = \frac{|\rho^{PF} - \rho^M|}{\rho^{PF}} \times 100$$

$$\eta = 1 - \frac{|\rho^{PF} - \rho^M|}{\rho^{PF}}$$

### 3. Using Aquami software

This package, written in python, is used for microstructure analysis such as bright field area, dark field area, ligament diameter etc.

We have used Aquami over Phase field generated image and Machine generated image.

## Results & Discussions

The training and testing loss curve is shown in Figure 2 as a function of number of epoch. It can be observed that the training as well as testing loss is decreasing with the number of epochs. This clearly reflects that the network is adjusting its weights and biases through learning process after each epoch.

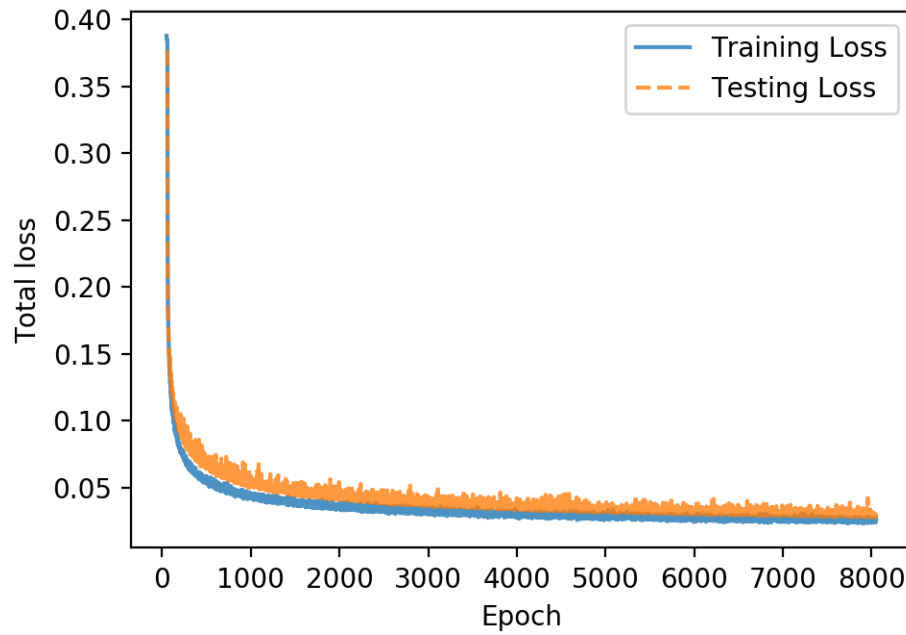


Figure 2: Training and testing loss as a function of number of epoch.

The result of the three methods that we have done in order to calculate the efficiency of machine learning algorithm is discussed below:


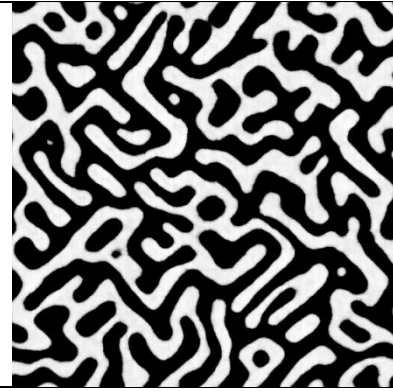
#### Particle analysis (using composition data)

We have discussed earlier that the composition data is scaled between -1 and 1 where -1 represents dark pixel while 1 represents bright pixel. We have done an analysis for calculating the amount of the precipitate and the matrix for both the library image and the machine generated image to quantify the efficiency of machine generated image. The negative part (from -1 to 0) represents matrix while the positive part (from 0 to 1) represents the precipitate. The calculated percentage of matrix and precipitate are in good agreement for a particular microstructure for both library and the machine generated one (see Figure 3).

#### Particle analysis (using ImageJ software)

#### TRAINING



c110: 501, c120: 208, c440: 117, c111: 0, c121: 0, c441: 0, xs = 0.0, ys = 0.0

	Phase Field	Machine	%Error	Average %Error
				<b>4.572</b>
%Matrix	54.4	56.668	4.169	
%Precipitate	45.6	43.332	4.974	

$$\eta = 0.9602$$

#### TESTING

c110: 742, c120: 336, c440: 96, c111: 0, c121: 0, c441: 0

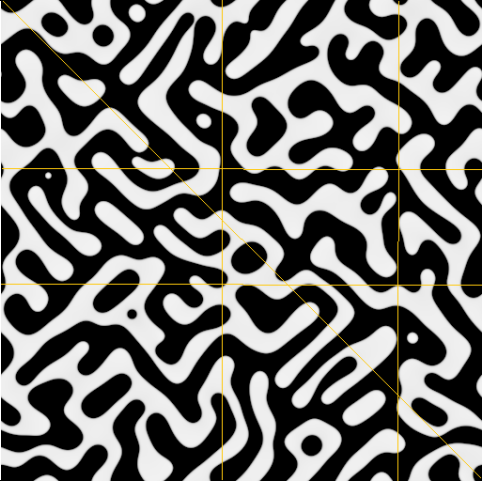
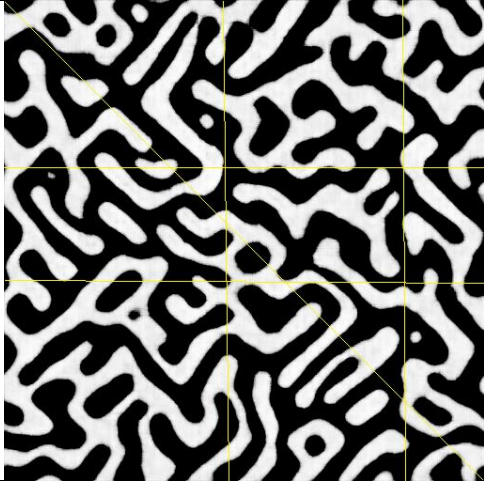
	Phase Field	Machine	%Error	Average %Error
				<b>4.862</b>
%Matrix	55.335	57.738	4.343	
%Precipitate	44.665	42.262	5.380	

$$\eta = 0.9502$$

Using the random lines method.

### TRAINING

c110: 501, c120: 208, c440: 117, c111: 0, c121: 0, c441: 0

Phase Field		Machine	
			
$l_{avg}^{PF}$	551.05	$l_{avg}^M$	552.28
$\bar{l}_{avg}^{PF}$	19.3	$\bar{l}_{avg}^M$	19.4

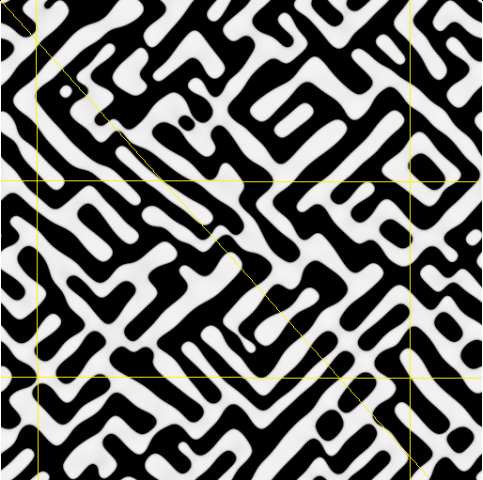

$\rho^{PF}$	0.03502	$\rho^M$	0.03513
-------------	---------	----------	---------

**%Error = 0.314**

**$\eta = 0.9969$**

## TESTING

c110: 742, c120: 336, c440: 96, c111: 0, c121: 0, c441: 0

Phase Field		Machine	
			
$l_{avg}^{PF}$	546.04	$l_{avg}^M$	545.68
$i_{avg}^{PF}$	22.1	$i_{avg}^M$	22.3
$\rho^{PF}$	0.04047	$\rho^M$	0.04087

**%Error = 0.988**

**$\eta = 0.9901$**

The efficiency scores obtained using each of the above discussed model is represented in a bar plot as shown in Figure 3.



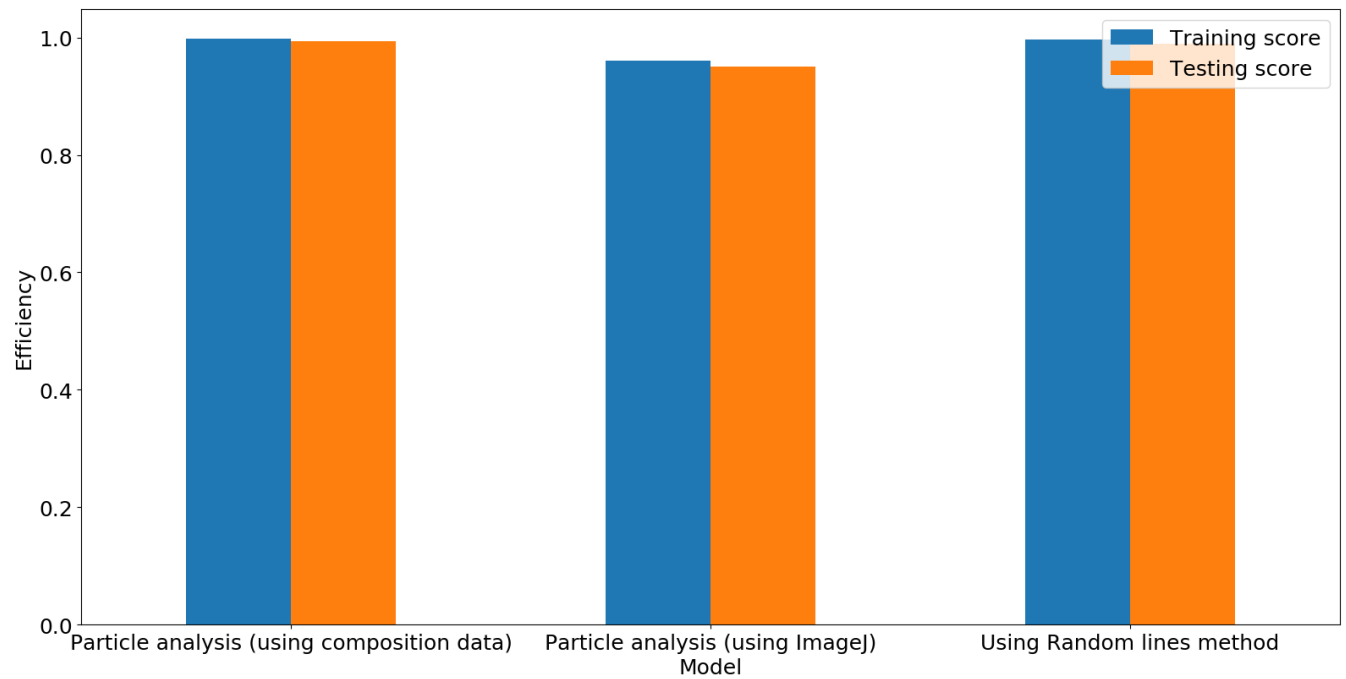


Figure 3: The efficiency of different methods used for analyzing the quality of machine generated image.

## Receiver Operating Characteristic (ROC) curve

Further, the efficiency of machine generated image is quantified using ROC curve which is generally used in statistical modeling to study the accuracy of statistical models. The ROC curve is the sensitivity as a function of fallout. Since the composition of the material in our dataset varies between -1 and 1 (where -1 to 0 represents precipitate or second phase while 0 to 1 represents matrix), we represent the negative composition with 0 and positive composition to 1 (including 0). The ROC curve for a particular training and testing microstructure is shown in Figure 2 and 3 respectively.

**Training** (c110 = 1064, c120 = 532, c440 = 266, c111 = c121 = c441 = 0, x-strain = y-strain = 0, t = 1800)

**Accuracy Score = 97.34%**

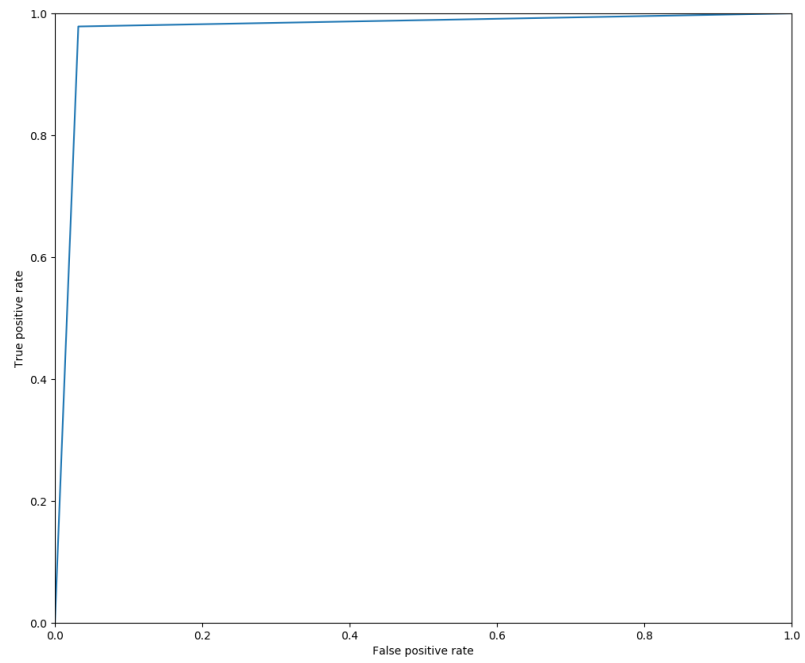


Figure 2: ROV curve for a training microstructure dataset.

**Testing** ( $c_{110} = 1000$ ,  $c_{120} = 462$ ,  $c_{440} = 269$ ,  $c_{111} = c_{121} = c_{441} = 0$ ,  $x\text{-strain} = y\text{-strain} = 0$ ,  $t = 1800$ )  
**Accuracy Score = 92.75%**

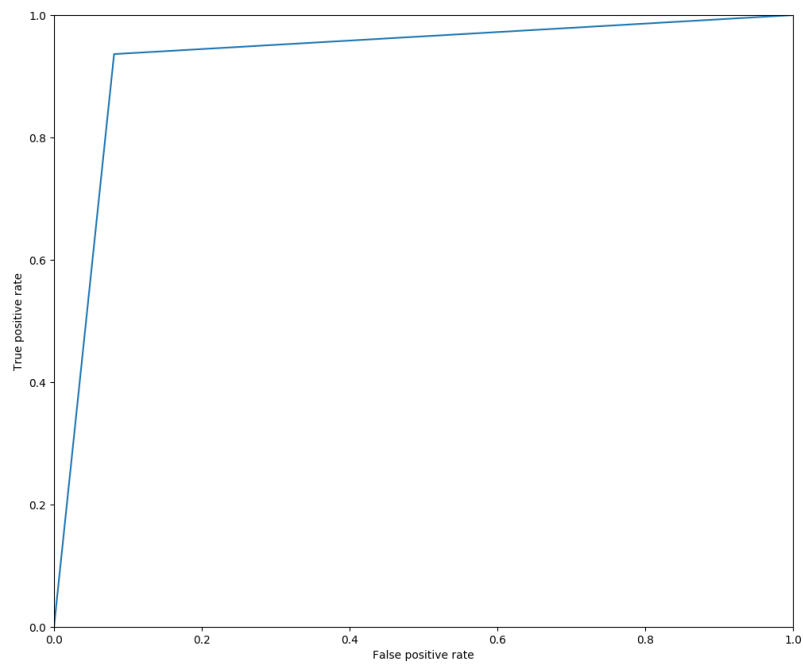




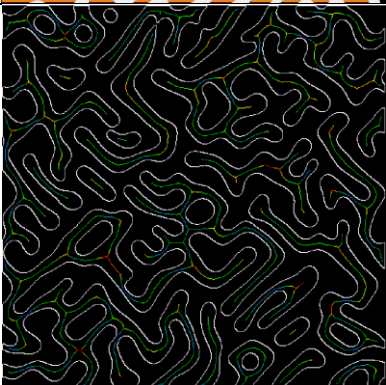



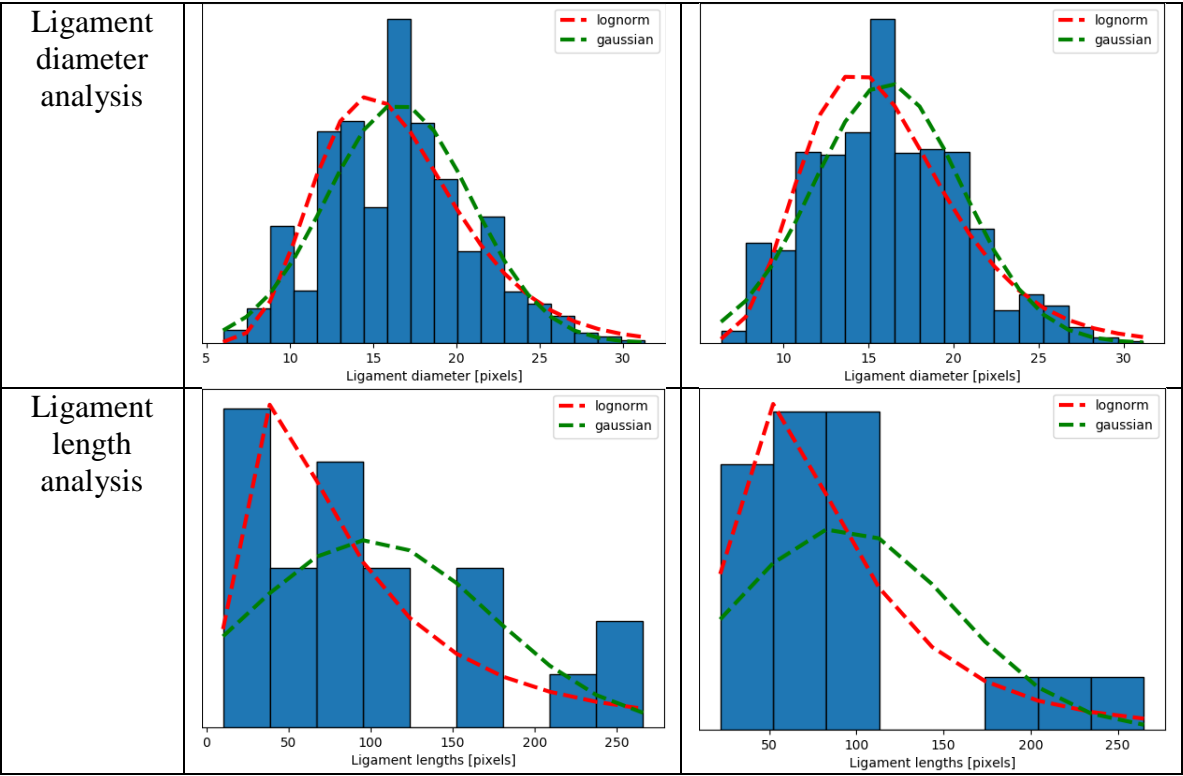
Figure 3: ROC curve for a testing microstructure dataset.

Using Aquami software

TRAINING

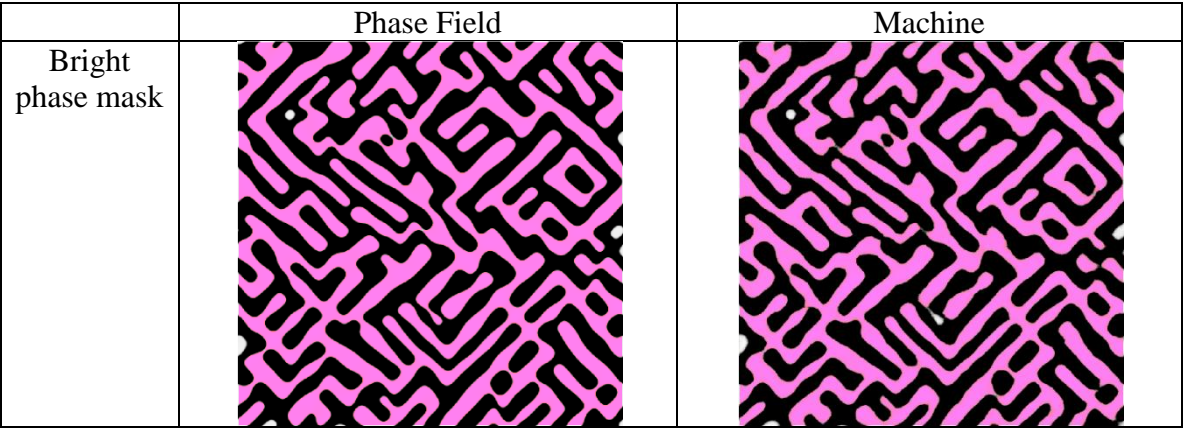
c110: 501, c120: 208, c440: 117, c111: 0, c121: 0, c441: 0

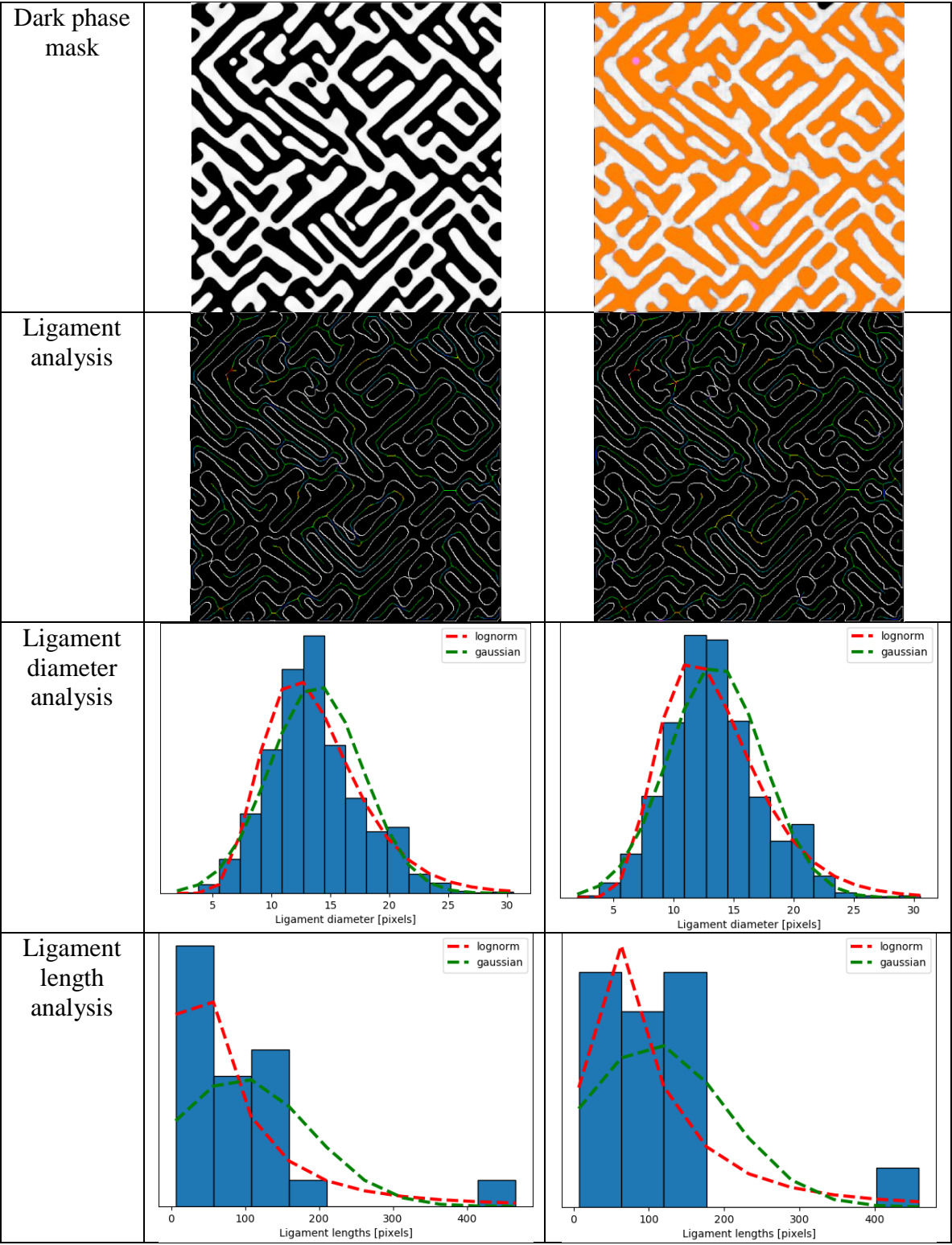
	Phase Field	Machine
Bright phase mask		
Dark phase mask		
Ligament analysis		



**TESTING**

c110: 742, c120: 336, c440: 96, c111: 0, c121: 0, c441: 0





The ligament length and diameter analysis by Aquami software produce similar curves for the microstructures generated through phase field and machine learning. This further reflects good efficiency of the neural network model.

## Unsupervised learning

### 1. Agglomerative clustering

```
cluster = AgglomerativeClustering(n_clusters = 72, affinity = 'euclidean').fit(dataset1)
```

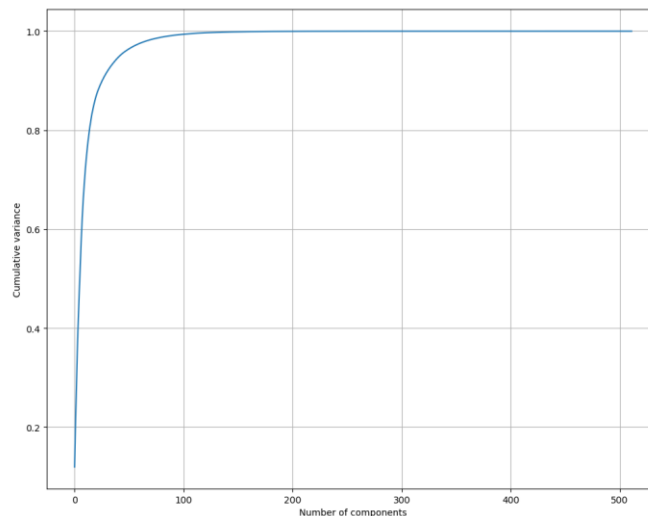
```
Print (cluster.labels_)
```

```
[11 11 11 13 13 13 13 13 13 13 13 45 45 45 45 45 45 45 49 49 49 49 49 49 65 65 65 65 65 33 33 33 33 33 66 66 66 66
32 32 32 32 32 32 32 32 38 38 38 38 38 38 38 64 64 64 64 64 42 42 42 42 42 70 70 70 31 31 31 31 31 31 62 62 62 62
62 25 25 25 25 25 25 3 3 3 3 3 3 3 50 50 50 50 50 59 59 59 59 59 59 67 67 67 67 21 21 21 21 21 21 21 2 2
2 2 2 2 2 2 35 35 35 35 35 35 35 60 60 60 60 60 60 39 39 39 39 39 39 40 40 40 40 40 40 12 12 12 12 12 12
58 58 58 58 58 48 48 48 48 48 48 23 23 23 23 23 23 71 71 71 53 53 53 53 10 10 10 10 10 10 16 16 16 16 16
16 16 1 1 1 1 1 1 1 1 1 47 47 47 47 47 47 6 6 6 6 6 6 24 24 24 24 24 8 8 8 8 8 8 5 5 5 5 5
5 5 19 19 19 19 19 19 19 7 7 7 7 7 7 7 61 61 61 61 61 20 20 20 20 20 20 0 0 0 0 0 0 15 15 15 15
15 15 15 15 52 52 52 52 52 28 28 28 28 28 28 57 57 57 57 9 9 9 9 9 9 51 51 51 51 51 26 26 26 26
26 26 26 17 17 17 17 17 22 22 22 22 22 22 36 36 36 36 36 36 56 56 56 56 56 37 37 37 37 37 37 34
34 34 34 34 34 30 30 30 30 69 69 69 4 4 4 4 4 4 55 55 55 55 14 14 14 14 14 14 46 46 46 46
46 46 46 63 63 63 63 27 27 27 27 27 27 44 44 44 44 44 29 29 29 29 29 41 41 41 41 41 18 18 18 18
18 18 18 43 43 43 43 43 68 68 68 54 54 54 54 11 11 11 11]
```

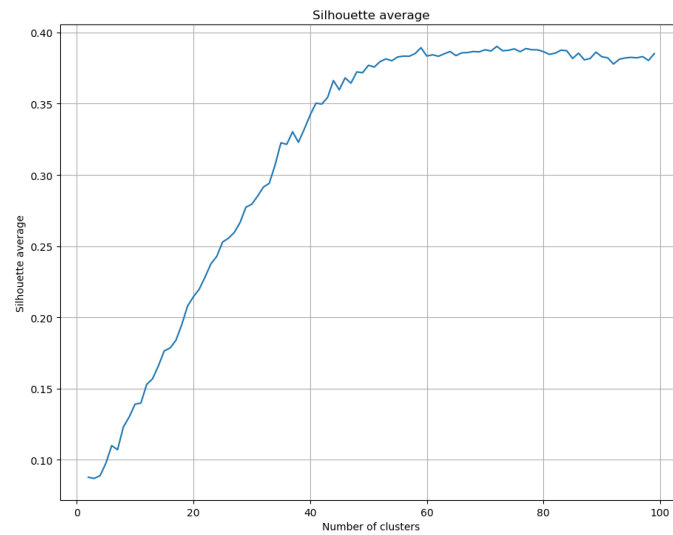
### 2. Principle component analysis

```
pca = PCA().fit(dataset1)
```

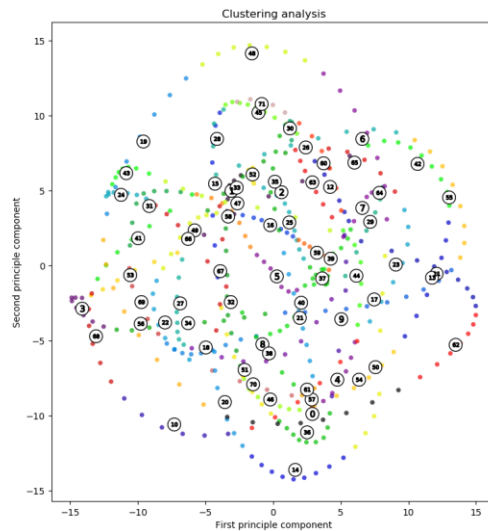
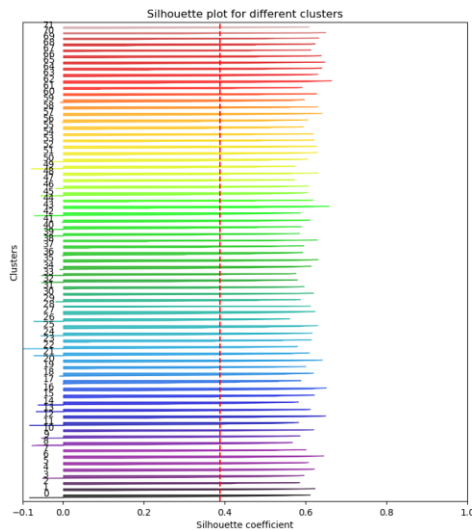
```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```



### 3. K-Means clustering

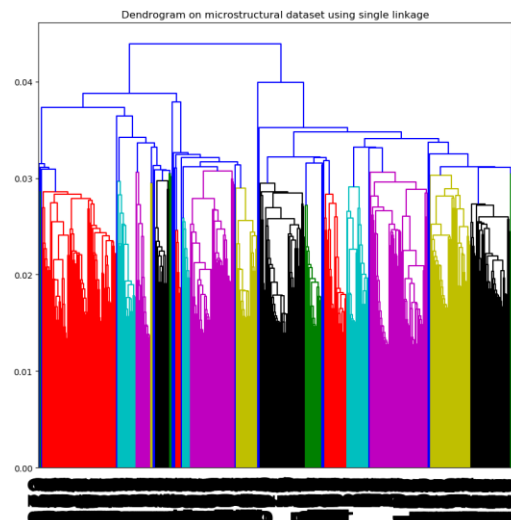
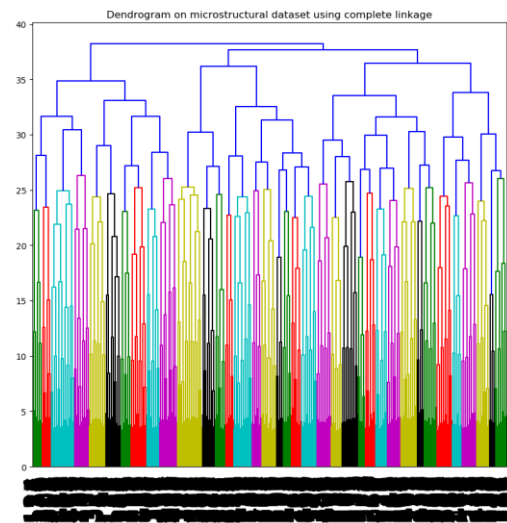
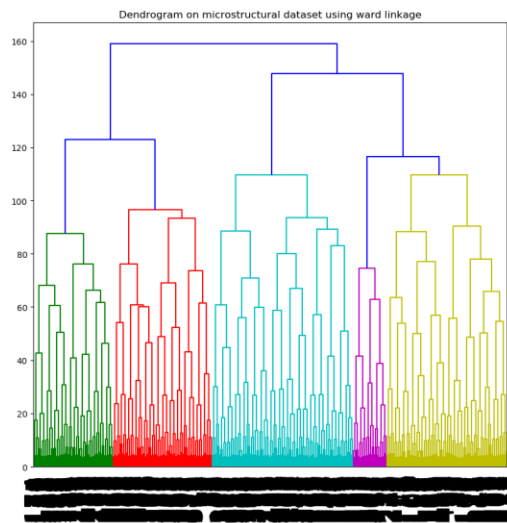


**Silhouette analysis for KMeans clustering on the microstructural dataset with  $n\_clusters = 72$**



#### 4. Hierarchical clustering

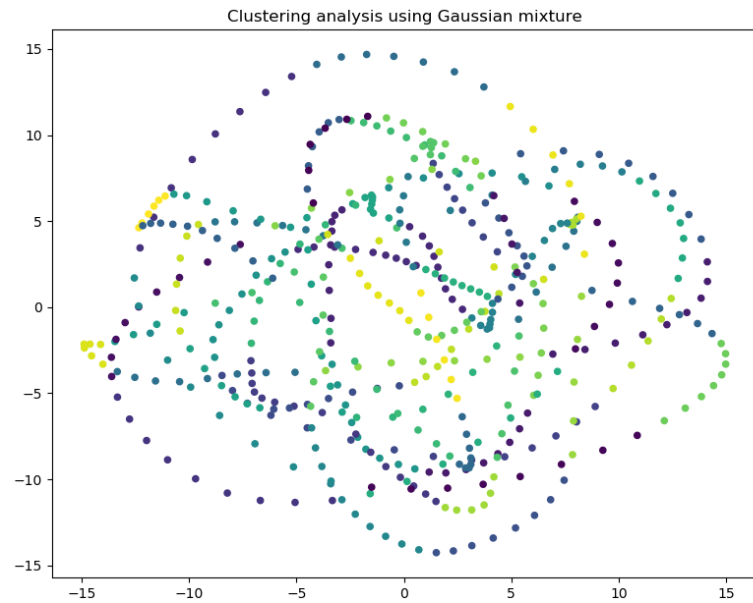
```
h = linkage(dataset1, 'ward')
dendro = dendrogram(H, leaf_font_size = 30)
plt.show()
h = linkage(dataset1, 'complete')
dendro = dendrogram(H, leaf_font_size = 30)
plt.show()
h = linkage(dataset1, 'single', metric = 'correlation')
dendro = dendrogram(H, leaf_font_size = 30)
plt.show()
```



## 5. Gaussian mixture

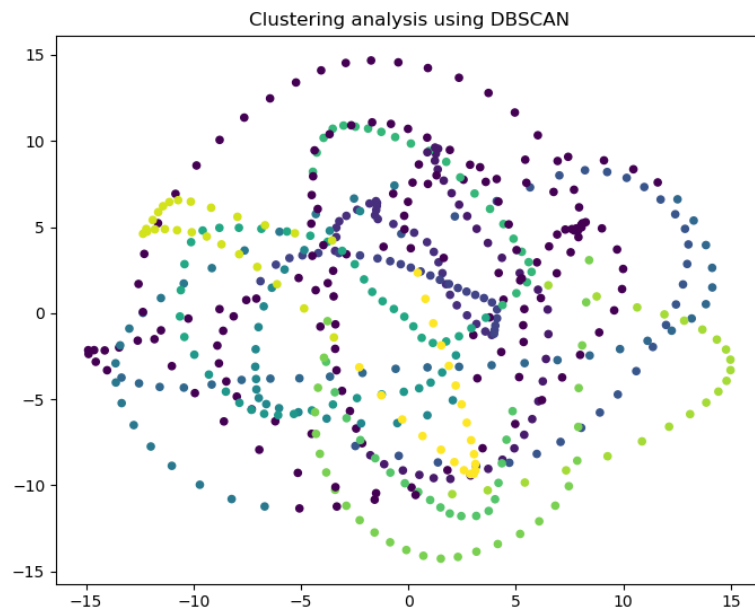
```
g_m = GaussianMixture(n_components = 72).fit(x)
plt.scatter(x[:, 0], x[:, 1], c = labels, s = 20, cmap = 'viridis')
plt.show()
```





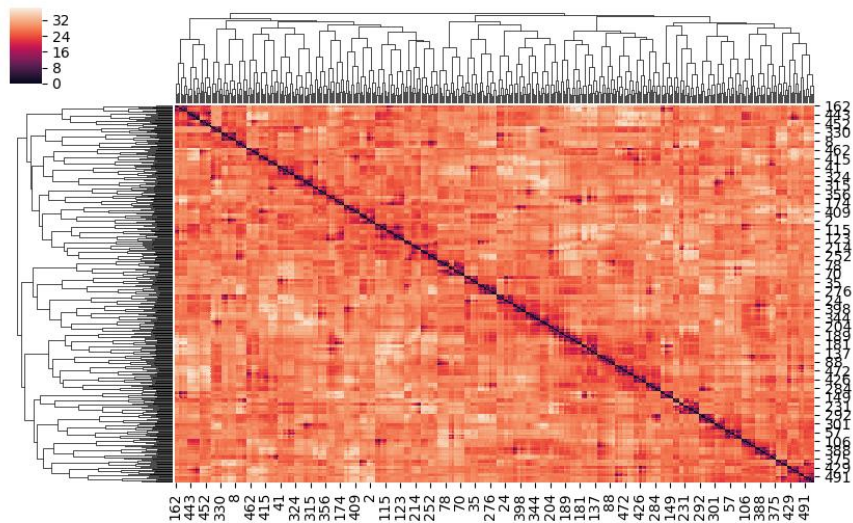
## 6. DBSCAN

```
db = DBSCAN(eps = 0.3, min_samples = 10, metric = 'cosine').fit(x)  
plt.scatter(x[:, 0], x[:, 1], c = labels, s = 20, cmap = 'viridis')
```

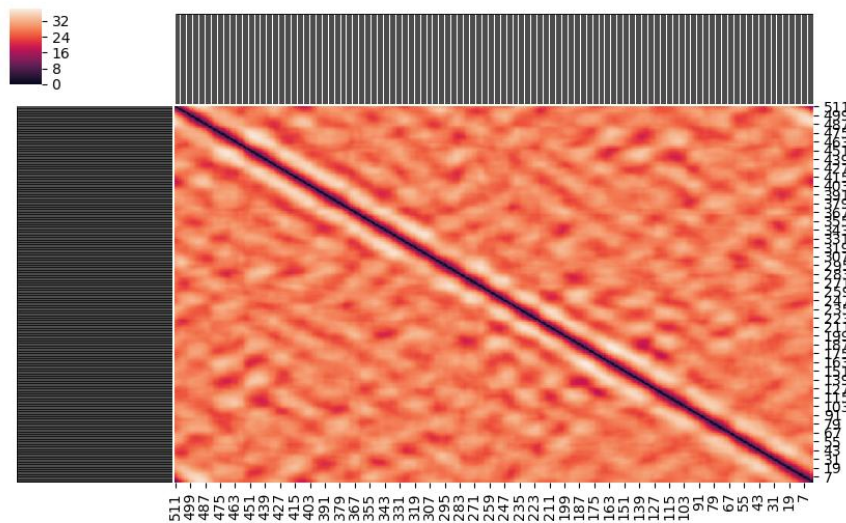


## Analysis using Distance matrix and Heat maps

```
dm = squareform(pdist(dataset1))
h = sns.clustermap(dm, metric = 'euclidean')
plt.show()
```



```
h = sns.clustermap(dm, metric = 'jaccard')
plt.show()
```

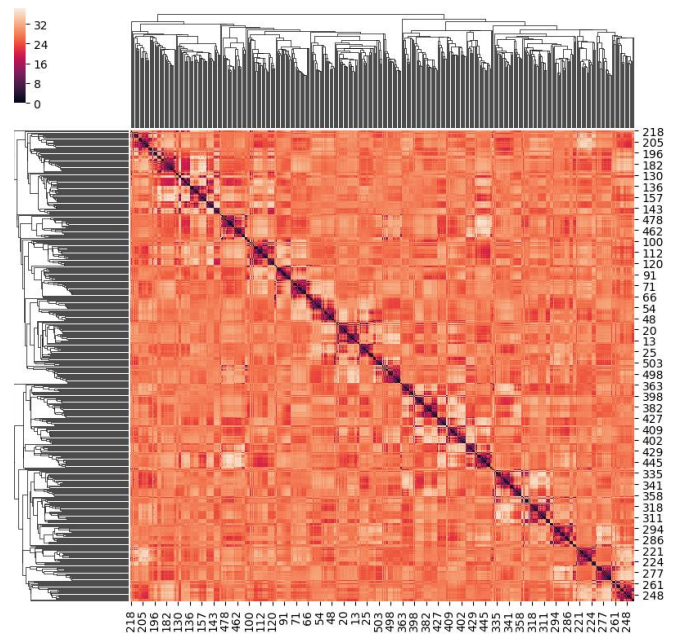
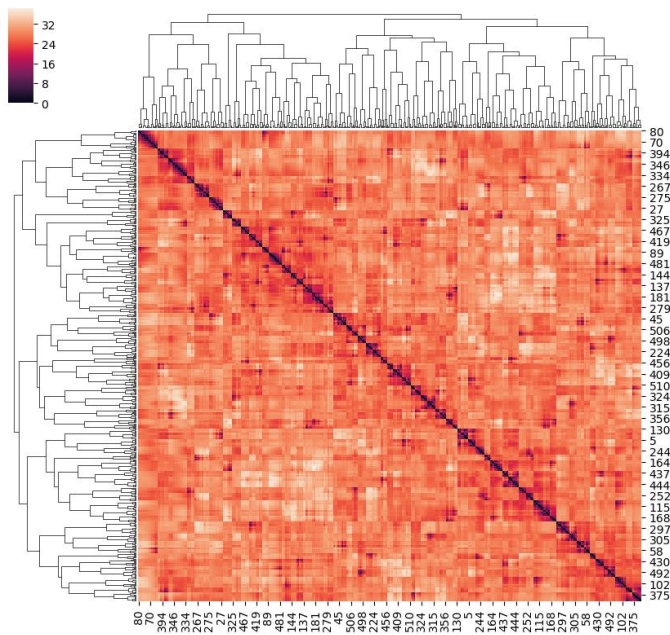


```
h = sns.clustermap(dm, metric = 'correlation')
```

```
plt.show()
```

```
h = sns.clustermap(dm, metric = 'single')
```

```
plt.show()
```



## References

- [1] Jingzhi Zhu et al 2001 *Modelling Simul. Mater. Sci. Eng.* 9 499
- [2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “*Gradient-based learning applied to document recognition*,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [3] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “*Handwritten digit recognition with a backpropagation network*,” in *Advances in neural information processing systems*. Citeseer, 1990.
- [4] Khachaturyan A G 1983 *Theory of structural transformations in Solids* (New York: Wiley)
- [5] Cahn J W 1961 *Acta. Metall.* 9 795