Application of neural network for nano-scale image classification of materials

[1]Srija Biswas

[1]Department of Materials Science and Engineering, Stanford University | srijab97@stanford.edu

1. Abstract

The ability to identify and recognise specific features within images is of particular interest to scientists working with microscopy techniques. Training a neural network on SEM images would provide many advantages: (i) automatic image classification; (ii) a searchable database which allows scientists to find a specific category of SEM images; (iii) potential for feature extraction to accomplish specific tasks[1]. This study uses the possible architectures as mentioned in literature, like Inception V-3 and explores well known architectures like DenseNet that yields a high accuracy of about 92% after training for 100 epochs. A new architecture using AutoEncoders has been tried along with DenseNet that can achieve over 90% accuracy.

2. Introduction

Neural networks have been employed for machine learning in a number of recent studies[2,3], extracting features from different types of microscope images. Image recognition techniques can be a very powerful tool in nanoscience, where a large number of images are the typical outcome of characterization techniques such as scanning electron microscopy. A scanning electron microscope (SEM) is a versatile instrument which is routinely used in nanoscience and nanotechnology to explore the structure of materials with spatial resolution down to 1 nm. The aim of this project is to apply various transfer learning methods to do image recognition, automatic categorization, and labelling of materials surface morphology (in nano scale) based on images obtained by SEM. The ultimate goal would be to develop a versatile neural network in classifying the morphologies/structure as observed from SEM micro-graphs.

3. Related Work

The project is closely associated with the work by Modarres, M.H., Aversa, R., Cozzini, S. *et al.* Neural Network for Nanoscience Scanning Electron Microscope Image Recognition. *Sci Rep* **7**, 13282 (2017)[1]. This work serves as earliest concrete effort to use Deep Learning techniques in SEM Image classification and possibly the only one in this exact topic. The paper reports using pretrained Inception-v3 network and then fine-tuning the last layer and retrain the network allowing back-propagation through all the layers. They have also reported comparing Inception models trained from scratch. However, all the code used in the paper unavailable for use. It is reported that after 400 epochs the Inception-v3 model shows about 95.3% test accuracy(with feature extraction and fine tuning).

4. Dataset and Features

The major challenge would be defining a proper dataset that is mostly agreeable by all. As shown in Fig. 1, depending upon the magnification, orientation or the quality of the SEM images, the classification category may appear to vary. There also might be various morphologies present together. Thus, making the classification even more challenging. Dataset of 21,169 SEM images produced at CNR-IOM (Trieste, Italy) where images are classified into 10 categories in a folder structure. Classification labels have been checked by a group of materials scientists on the web site http://sem-

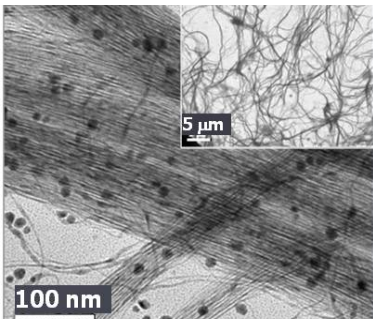


Fig. 1. SEM image showing different morphologies as seen under different magnification. Lower magnification shows nanowires while higher magnification shows particles in a wire matrix.[4]

classifier.nffa.eu and only those images which have been validated by the 100% of the group have been included in the dataset[5]. The entire dataset has been partitioned randomly (manually) in 80:20 ratio across each class: 'Biological', 'Fibres', 'Films_Coated_Surface', 'MEMS_devices_and_electrodes', 'Nanowires', 'Particles', 'Patterned_surface', 'Porous_Sponge', 'Powder', 'Tips'. All the images are of dimension 1024x768. Two important features of the data set include, (1) Hierarchical data and (2) Image taken at different magnifications. Examples of the following are illustrated in Fig. 2 (a,b) where both these images are categorized as **Fibres** but the features look very different for a person unfamiliar with nano-materials and related sciences. In larger magnification the image on right is a **Fibre** but in lower magnification the intertwining of fibres assumes a **surface** like morphology. Similarly, in Fig. 2 (c,d), both these images might look similar to a person unfamiliar in this domain. However, there is a scale bar on the bottom left corner that shows the left image is taken at a much lower magnification. The left image is classified as **Fibres** as it is in the macro-scale and the right image is that of a **nanowire**.

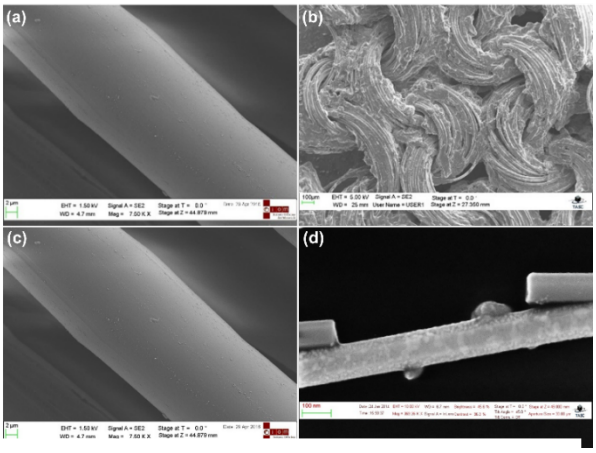


Fig. 2. SEM micrographs of materials categorized as : **(a, b, c)** "Fibres". and **(d)** "Nanowire".

5. Method

5.1. Baseline:

5.1.1. Training with a shallow 6-layer CNN from Scratch:

A 6-layer CNN (Fig. 3) has been trained from scratch[4]. Cross Entropy loss has been used as the loss function and Adam as the optimizing function (learning rate 0.001 and weight decay 0.0001).I normalized the inputs to 0.5 mean and 0.5 standard deviation across the channels. Both horizontal and vertical flips have been used in data transformation as all these operations translates as the direction of loading a sample in SEM that can be in any direction. Results are shown in Fig. 4.

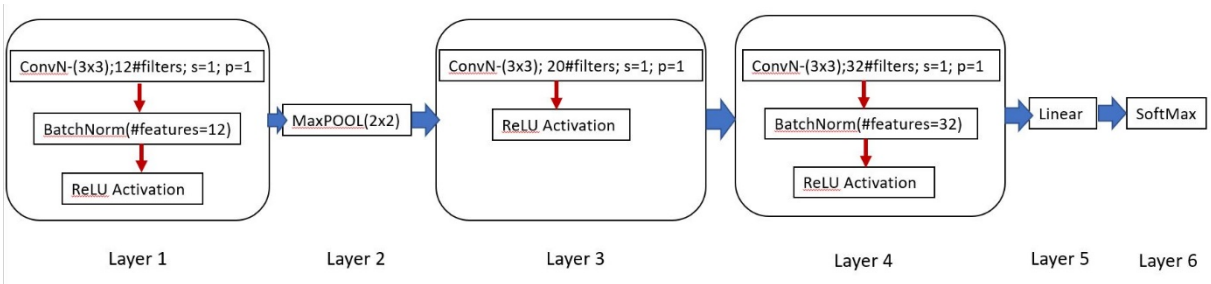


Fig. 3. Schematic flowchart representation of 6-layer CNN architecture.

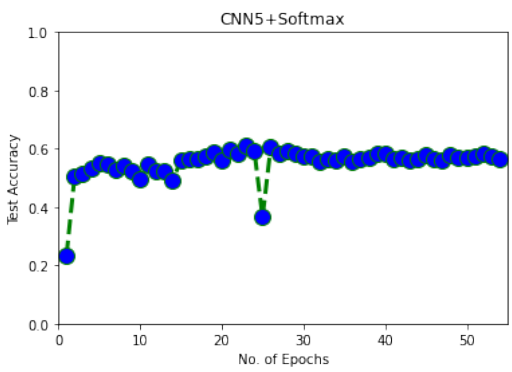


Fig. 4. Test accuracy of the architecture as a function of the timescale. Test accuracy is chosen as the optimising parameter [6]. After running for **55 epochs (~8 hrs)**, the **test accuracy** saturates close to **56.5%.** The training loss in the first epoch is 71.45 which is decreased to a value of 0.026 at the end of the 55th epoch.

5.1.2. Transfer Learning with a pre-Trained Inseption-v3 model (while retaining the last fully connected layer and including the auxiliary logits):

In this part, transfer learning was tested on our target SEM data set, using Inception-v3. This approach is faster than training from scratch, but the results might be less accurate in some cases (e.g., feature extraction), and the architectures which can be used are restricted to the pre-trained checkpoints available in the literature. The model has been pretrained to an Imagenet Database and applied to the dataset. The paper reports with pretraining they can get accuracy up to 90% after training for 160 epochs. However, the source code is unavailable. So I have used an available code from Github[7][8][9]. I chose the optimising function as Adam(lr=0.005 and weight decay =0.0001).I tuned the data loaders and transformers according to the needs of this project. The total loss was taken to be a combination of loss from output and aux logits (Cross Entropy loss was evaluated individually). The input images were taken in full resolution : 1024 x 768. Results are elucidated in Fig. 5. (see figure caption).

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Table 1. Inception v3

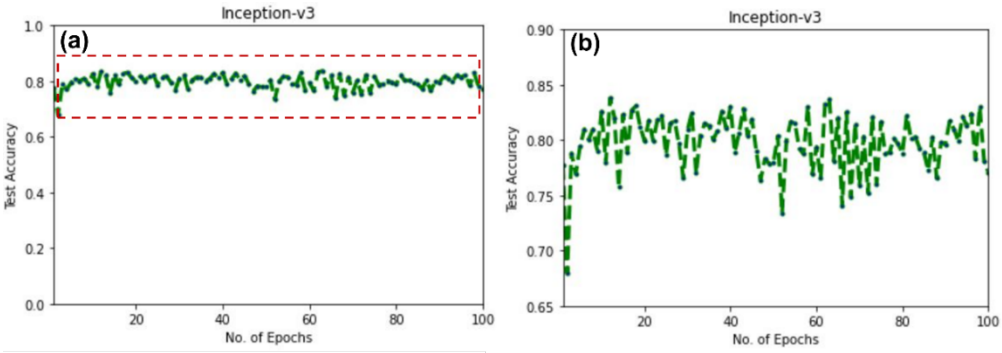
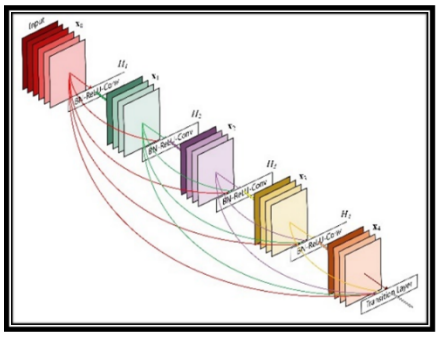


Fig. 5. The right image is the magnified version of the highlighted section of the image on left (Test Acc. Vs No of Epochs). **Test accuracy is ~82%** after training for 100 epochs (~24 hrs). Training loss improved from 2.77 to 1.19.

5.2. DenseNet121:[Application of old architecture in new problem]

DenseNet[10] is a network architecture where each layer is directly connected to every other layer in a feed-forward fashion (within each dense block). For each layer, the feature maps of all preceding layers are treated as a separate input whereas its own feature maps are passed on as inputs to all subsequent layers. DenseNet architecture as it is known to handle the different magnification problem of the dataset considered. Fig. 6. Illustrates the architecture of DenseNet121.



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201
Convolution	112 × 112	7 × 7 conv, stride 2		
Pooling	56 × 56	3 × 3 max pool, stride 2		
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv		
	28 × 28	2 × 2 average pool, stride 2		
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv		
	14 × 14	2 × 2 average pool, stride 2		
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Transition Layer (3)	14 × 14	1 × 1 conv		
	7 × 7	2 × 2 average pool, stride 2		
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32
Classification Layer	1 × 1	7 × 7 global average pool		
		1000D fully-connected, softmax		

Fig. 6. (Left) A dense block with 5 layers and growth rate 4. (Right) Table elucidates architecture of DenseNet121[11].

5.2.1. Model Parameters (for pretrained and trained from scratch DenseNet121:The model parameters are as follows : *(a) learning rate : 0.005, (b) weight decay : 0.0001, (c) batch size : 32, (d) image size : 512 x 512, (e) loss function : cross entropy loss, (f) optimizing function : Adam, and (g) data augmentation : random horizontal flip.*

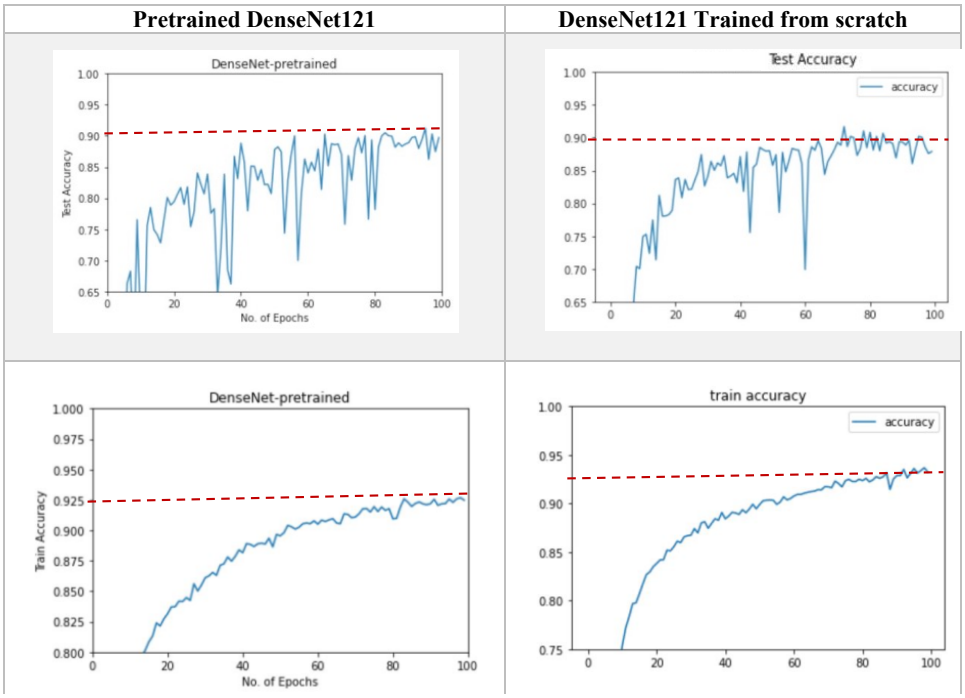


Fig. 7. Pretrained vs trained from scratch DenseNet121. Top two images depict train accuracy and the bottom two depict test accuracy. **Test accuracy** crosses **90%** for DenseNet trained from scratch (M0) and shows a relatively smooth graph (less noisy) when compared with pretrained DenseNet model. **Train Accuracy** reaches slightly above **92.5%** for the model trained from scratch (M0) after 90 epochs.

5.2.2. Hyperparameter Tuning: (Results after running 100 epochs)

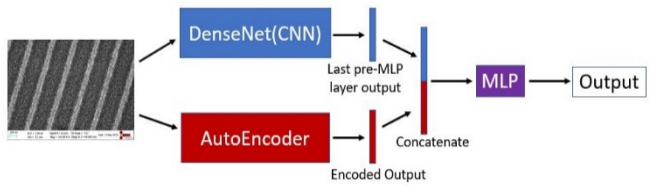
H= Horizontal Flip
V = Vertical Flip

	Model Type	Data Augmentation	Image Size	Learning Rate	Weight Decay	Optimizer	Train Accuracy	Test Accuracy
M1	DN-121(scratch)	Training set(H)	300 x 300	0.005	0.0001	Adam/Cross Entropy	~88%	~86%
M2	DN-121(scratch)	Training set(H+V)	512 x 512	0.01	0.0001	Adam/Cross Entropy	~87%	~84%
M3	DN-121(scratch)	Training set(H)	512 x 512	0.02	0.0001	Adam/Cross Entropy	~88%(very noisy data)	~85%(very noisy data)
M4	DN-121(scratch)	Training set(H)	600 x 425	0.007	0.0001	Adam/Cross Entropy	~87%	~85%

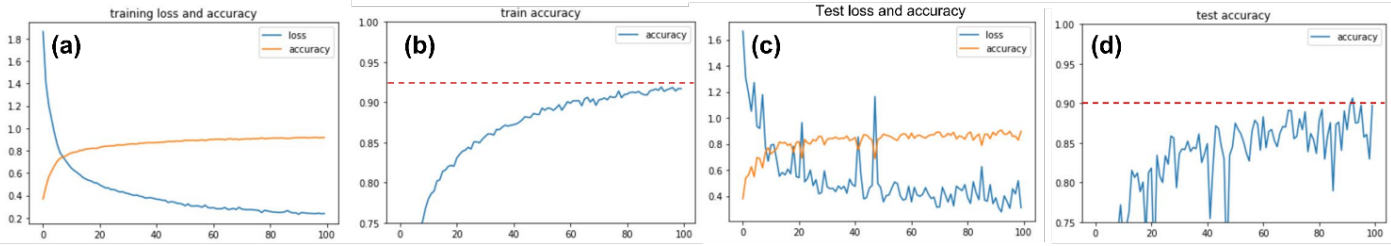
Reducing the image size (keeping other parameters constant) as seen in M1, makes the training faster but at the end the train and test accuracy are considerably lower. Increasing the learning rate to 0.01 as in M2, during the initial epochs the increase in test/train accuracy happens faster than M0 but after about 70 epochs the change is very slow and more random as the values all tend to oscillate around a point. More Data augmentation (vertical rotation) on the training set has little effect in boosting the accuracy. Using the same data augmentation as in M0 but increasing the learning rate as in M3, makes

the accuracy curves very noisy and difficult to obtain a steady value. Running more epochs is required to get a stable value. In M4 the image sizes were changed so that the aspect ratio is same as the original image. The learning rate was slightly increased, but the test accuracy remained lower, possibly indicating that square resized image work better.

5.3. Auto Encoder + DenseNet121: [Application of new architecture in new problem]



The main motivation to use an auto encoder along with the DenseNet is that a CNN captures the local features of an image whereas using an auto encoder with fully connected layer effectively captures the global features. The Auto encoder (*schematic displayed left*) condenses the features of the image into a dense featured output which can be concatenated with the pre-MLP output of the Densenet network, where the input now is expected to have a better trade-off between global and local feature representation.



(e)	Model Type	Data Augmentation-	Image Size	Learning Rate	Weight Decay	Optimizer	Train Accuracy	Test Accuracy
	AutoEncoder +DN-121(scratch)	Training set(H)	480 x 480	0.005	0.0001	Adam/Cross Entropy	~92%	~90%
	AutoEncoder +DN-121(scratch	Training set(H)	128 x 128	0.005	0.0001	Adam/Cross Entropy	~89%	~87%
	AutoEncoder +DN-121(scratch	Training set(H)	480 x 480	0.01	0.0001	Adam/Cross Entropy	~87%	~86%(very noisy data)

Fig. 8. The loss & accuracy plots of train (a,b) and test (c,d) dataset for AutoEncoder + DenseNet model- AD1 for 100 epochs(~20 hrs). (**Table e.**)

When the input image size is reduced the train/test accuracy reduces as in AD2 when compared to AD1. The processing time for each epoch reduces by half the value of AD1. Increase in learning rate to 0.01 and keeping all other hyperparameters same as AD1, the test accuracy has a lot of noise and does not appear to converge very well as the model trains to higher epochs.

6. Future work/Conclusion

DenseNet network proves to be very efficient in classifying SEM images. In comparison to the work by Modarres et al[1], DenseNet architecture gives slightly better performance when compared to the model presented at the 100 epoch limit. The Auto Encoder + DenseNet architecture gives great result but further tuning is necessary and for that higher GPU power is required to carry out the training for full resolution input images. I also wish to try transformer based autoencoders to understand how the global features are captured and represented.

7. Reference

[1] Modarres, M.H., Aversa, R., Cozzini, S. *et al.* Neural Network for Nanoscience Scanning Electron Microscope Image Recognition. *Sci Rep* **7**, 13282 (2017). https://doi.org/10.1038/s41598-017-13565-z

[2]Nikiforov, M. P. *et al*. Functional recognition imaging using artificial neural networks: applications to rapid cellular identification via broadband electromechanical response. *Nanotechnology* **20**, 405708, http://stacks.iop.org/0957-4484/20/i=0/a=405708 (2009).

[3] Al-Khedher, M. A., Pezeshki, C., McHale, J. L. & Knorr, F. J. Quality classification via raman identification and sem analysis of carbon nanotube bundles using artificial neural networks. *Nanotechnology* **18**, 355703, http://stacks.iop.org/0957-4484/18/i=35/a=355703 (2007)

[4]https://chemistry.tau.ac.il/markovich/index.php/research?fbclid=IwAR2WBeY5Lgx6EWvtpo4w2P6TwQmGx9UmLk0KXelLMrQORCGLK4xWeS74G8Y#A1

[5] NFFA-EUROPE. Draft metadata standard for nanoscience data. NFFA project deliverable D11.2, http://www.nffa.eu/media/124786/d112-draft-metadata-standard-for-nanoscience-data_20160225-v1.pdf (2016).

The dataset is appropriate for the purposes of this study and in general for visual object recognition software research. Any scientific metadata associated to the measure is not present in the images. The dataset is therefore relevant as a whole, being the single images entirely detached from any specific information or scientific detail related to the displayed subject. This work has been done within the **NFFA-EUROPE** project (www.nffa.eu) and has received funding from the **European Union's Horizon 2020 Research** and Innovation Programme under grant agreement No. 654360 NFFA-Europe.

[6]Partially Inspired (last layer added by me and dataset tuning with optimizer selection) from: https://github.com/gaurav67890/Pytorch_Tutorials/blob/master/cnn-scratch-training.ipynb

The Inception-v3 model has been a combination of the github urls below but I have changed the optimizing functions/loss functions and hyperparameters as well as data transformation myself:

[7] https://github.com/devangsharma14/Dog-Breed-Classifier/blob/main/dog_app.ipynb

[8] https://github.com/Harry24k/Pytorch-Basic/blob/master/Week5/20.%20Transfer%20Learning%20with%20Inception%20v3.ipynb

[9] https://github.com/vatsmanish/Inception-v3-with-pytorch/blob/master/InceptionV3FromScratch.ipynb

[10] https://github.com/ihamdi/Dogs-vs-Cats-Classification/blob/main/pytorch-cat-vs-dog.ipynb

[11] https://github.com/liuzhuang13/DenseNet