
MATERIAL DATABASE CLASSIFICATION
USING CNN ARCHITECTURE
GROUP LEVEL - B

AUTHORS

JHUMA MIM

12.2022

Contents

1	Introduction	2
1.1	Problem Statement	2
1.2	Objectives	2
2	Feature Extraction	2
3	Data Processing	2
3.1	Our dataset Balance or Imbalance?	3
3.2	Do we need data augmentation?	3
3.2.1	Dimension reduction	3
3.2.2	Flipping (vertical, horizontal)	5
3.2.3	Rotation	6
3.2.4	Grayscale (blue scale, red scale, green scale)	6
4	Models	8
4.1	Basic CNN	8
4.2	CNN with pre-trained imageNet	8
4.3	CNN with pre-trained ResNet50	12
5	Results and Model evaluation	12
6	Conclusion	15

1 Introduction

In this work, we used the material image database [1] that has operated for studying human material categorization for training and testing material recognition systems. This database was constructed with the specific purpose of capturing a range of real-world appearances of everyday materials (e.g., glass, plastic, etc.). Each image in this database (100 pictures per category, ten categories) was selected manually from Flickr.com to ensure various illumination conditions, compositions, colors, texture, and material sub-types.

1.1 Problem Statement

Compared to human performance in the original Flickr Material Database (FMD) pictures, the performance of present vision systems falls short. The proposed approach is developed, and the accuracy attained is comparable to that of humans on databases such as FMD, allowing the materials to be recognized.

1.2 Objectives

The objective is to be able to classify the data from 10 classes, namely Fabric, Foliage, Glass, Leather, Metal, Paper, Plastic, Stone, Water, and Wood, by using Convolution Neural Networks (CNN) as a baseline and finally compared with a more advanced approach like CNN with pre-trained model GoogleNet and ResNet-50. Different tasks will be done, starting from data pre-processing to the classification model to achieve the goal. To improve the accuracy of recognition of materials over Flickr Materials Database. The tasks will be processed in the way they follow the requirements of the approach.

2 Feature Extraction

It is the process of taking raw data and extracting valuable features for modeling. With images, this usually means extracting things like color, texture, shape, etc. We will do a different kind of dataset processing to improve the features as below. Finally, during training, we think that we will use extracting features such as GoogleNet(Inception-V3), VGGNet-19 ResNet-50 networks; these networks were pre-trained on the enormous data set known as the ImageNet dataset used in image classification tasks.

3 Data Processing

Before starting the experiment, we read the dataset and have provided labels for each category from(0 to 9).



Figure 1: Showing original dataset each category of the image from 0 to 9

3.1 Our dataset Balance or Imbalance?

We notice our dataset is a balanced class which means each class has the same number of images. We have 10 classes, and each class has 100 images.

3.2 Do we need data augmentation?

Though our dataset is balanced. We see that we have only 1000 samples; therefore, we experimented with augmenting the dataset. After data augmentation, our best model performance improved by 0.5% (Table 1). There are several works that mention model performance improvement after data augmentation as well [2]. Some data augmentation we have placed are examples, Reshaping images to lower dimensions, Flipping (vertical, horizontal), Rotation, and Grayscale conversion.

3.2.1 Dimension reduction

This is one of the image augmentation approaches. The image will be reduced in size and will help memory issues and faster runtime. Since our images are more than 500 px in size, therefore, we think a reduction of size will help memory issues and faster runtime. We experimented with different dimension reductions and found several image classifications done even with the smallest dimension $28 * 28$ dimensions [3]. However, since imageNet required $224 * 224$ and reducing too much can be a reason to lose essential features, therefore throughout our experiment, we have used this dimension.

Experiment name	Accuracy	Comments
Without data augmentation	75	Baseline results
Rotation Data augmentation	75.5	Accuracy improved 0.5 %
Flipping Data augmentation	75.3	Accuracy improved 0.3 %
Gray-scale conversion	73.3	Accuracy decreased 1.7 %
ADAM optimizer	75	No accuracy improvement
SGD optimizer	76	Accuracy improvement 1%
128 batch size	74	Accuracy decrease 1%
64 batch size	75	Same as baseline and we used it as optimum compared to 128 batch size
Leaning rate 0.001	75	Same as baseline%
Leaning rate 0.0001	75.2	Negligible improvement
Leaning rate 0.01	73.2	Accuracy decrease 1.8%
Rotation + Flipping + SGD + 64 batch size + 0.001 leaning rate	78.5	Accuracy increased 4.5%after applying all optimal parameters, and we have consider theses as optimum parameters in our experiment.

Table 1: Validation Accuracy result using different model parameter and data augmentation.

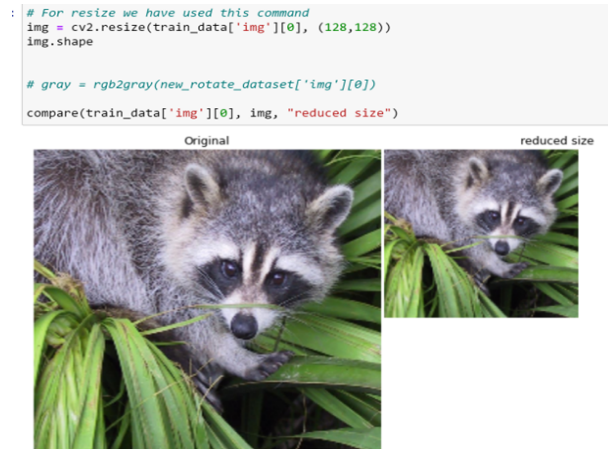


Figure 2: Example of image dimension reduction.

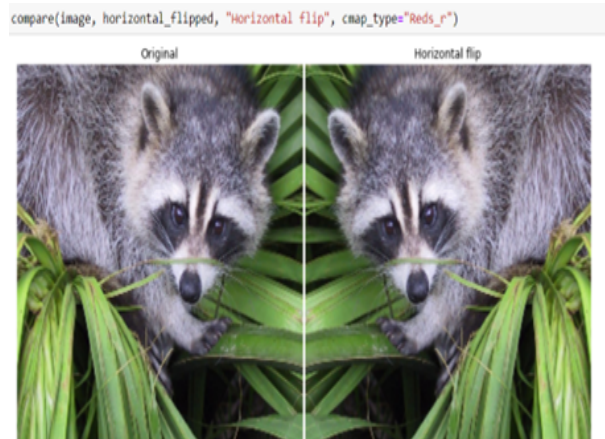


Figure 3: Showing flipped image

3.2.2 Flipping (vertical, horizontal)

The image will be converted into flip images horizontally and vertically. Some frameworks do not provide function for vertical flips. But, a vertical flip is equivalent to rotating an image by 180 degrees and then performing a horizontal flip.

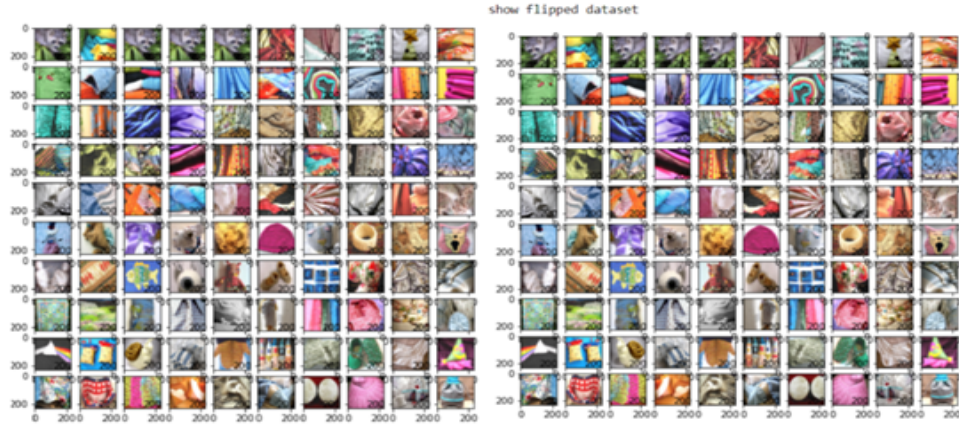


Figure 4: Original data vs flipped data

3.2.3 Rotation

This is yet another image augmentation technique. Rotating an image might not preserve its original dimensions (depending on what angle you choose to rotate it with).

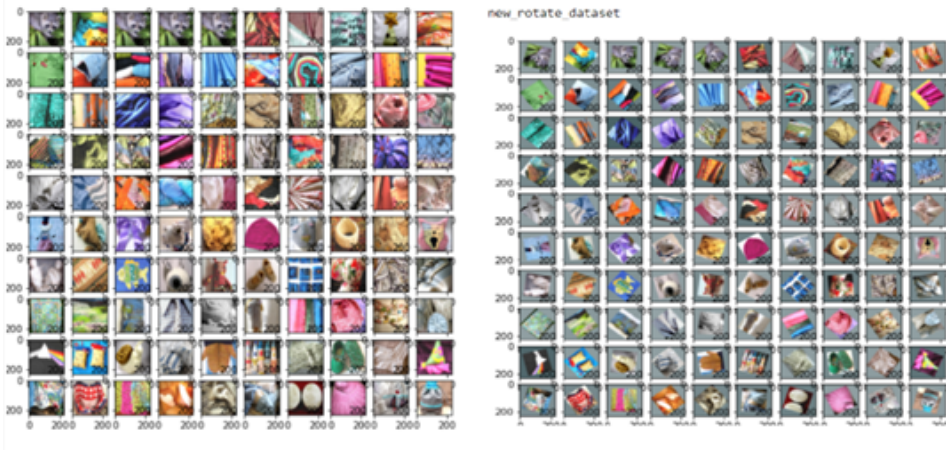


Figure 5: Original data vs flipped data

3.2.4 Grayscale (blue scale, red scale, green scale)

The image will be converted to grayscale the computer will assign each pixel a value based on how dark it is. All the numbers are put into an array, and the

computer does computations on that array. For imageNet, we have used RGB, and for basic convolution, we have used both RGB and Grayscale. Grayscale performed 1.7% less compared to taking whole RGB; therefore, all of our experiments used full RGB spectrum. The gray scale conversion might have lost some features.

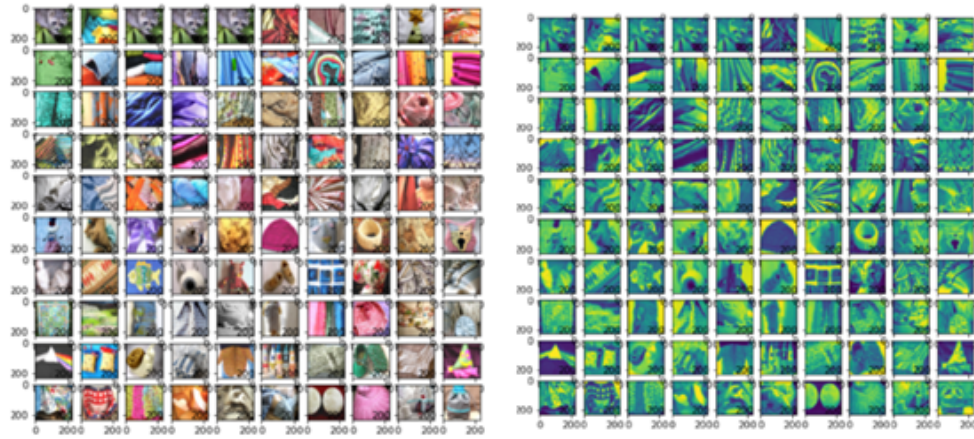


Figure 6: Original data vs grayscale

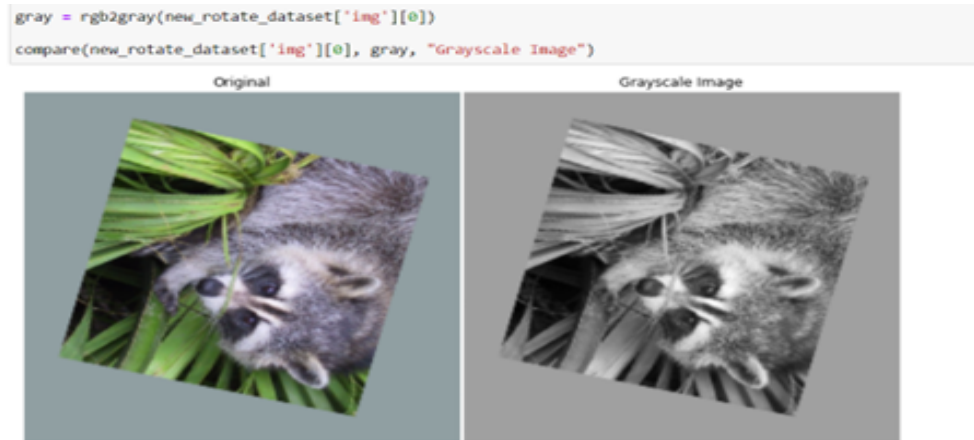


Figure 7: Rotate vs grayscale

4 Models

We used a total of three models in our experiments (i) Basic CNN, (ii) CNN with pre-trained imageNet (iii) CNN with pre-trained ResNet50. In all of our experiments, we have used the same preprocessing and data augmentation techniques so that we can compare final results.

4.1 Basic CNN

We have created a basic CNN model that has an Image Input Layer \rightarrow Convolution 2D Layer \rightarrow ReLU Layer \rightarrow Max Pooling Layer with 2D convolution Layer \rightarrow Fully Connected Layer with 10 Classes \rightarrow Softmax Layer and \rightarrow Classification Layer The final layer. The image input layers take $224 * 224$ dimension RGB image that we have used for imageNET and ResNe50. We used different filter sizes 8, 16, and 32, during the experiment. We have used shuffle in every epoch and validation iteration 3. We have tested with different layers and epochs (10 to 100) and optimizers (adam and sgdm). The best result of parameter tuning has mentioned in the results section.

4.2 CNN with pre-trained imageNet

GoogLeNet is type of convolutional neural network based on the Inception architecture. It makes use of Inception modules, which provide the network the ability to select from a variety of convolutional filter sizes in each block. An Inception network layers these modules on top of each other, with max-pooling layers with stride 2 on occasion to reduce the grid's resolution. The GoogLeNet Architecture is 22 layers deep, with 27 pooling layers included. There are 9 inception modules stacked linearly in total. The ends of the inception modules are connected to the global average pooling layer. GoogLeNet [4] architecture with the idea of having filters with multiple sizes that can operate on the same level. With this idea, the network actually becomes wider rather than deeper. The first version of the Inception network is referred to as GoogLeNet. Each layer in this design has a set convolution size. In the Inception module (1×1) , (3×3) , (5×5) convolution and (3×3) max pooling in parallel at the input, and the output of these is piled together to get the final result. The theory [4] is that convolution filters of various sizes will handle objects at various scales better. The architecture was designed to keep computational efficiency in mind. The idea behind that the architecture can be run on individual devices even with low computational resources. The architecture also contains two auxiliary classifier layer connected to the output of Inception (4a) and Inception (4d) layers. The architectural details of auxiliary classifiers as follows:

- An average pooling layer of filter size (5×5) and stride 3.
- A (1×1) convolution with 128 filters for dimension reduction and ReLU activation.

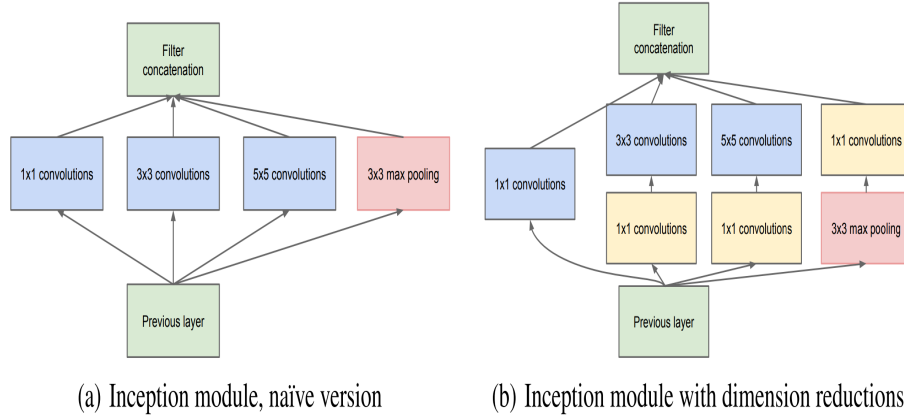


Figure 8: GoogLeNet incarnation of the Inception architecture

- A fully connected layer with 1025 outputs and ReLU activation Dropout Regularization with dropout ratio = 0.7
- A softmax classifier with 1000 classes output similar to the main softmax classifier.

This architecture takes image of size 224 x 224 with RGB color channels. All the convolutions inside this architecture uses Rectified Linear Units (ReLU) as their activation functions.

We use googlenet pretrained model with different normalization methods, we found it don't have obvious differences between those methods in the last week's workshop. So the method of zerocenter is used directly.

The optimizer of SGD with momentum is used. Compared with Adam, its advantage is that the consistency of the verification set and training set is better. At the same time, adma is more applied in NLP.


In classificationLayer, we change the output size to auto. And classes are changed to auto.

In fully-connected layer, values of WeightLearnRateFactor and BiasLearnRateFactor are set to 10, which can improve accuracy.

Others are still set according to the original structure of googlenet. GoogLeNet is a convolutional neural network that is 22 layers deep. The network trained on ImageNet classifies images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. Meanwhile it also has learned different feature representations for a wide range of images.




Figure 9: : GoogLeNet network with all the bells and whistles


classificationLayer ?

Name	<input type="text" value="classoutput"/>
Classes	<input type="text" value="auto"/>
ClassWeights	<input type="text" value="none"/>
OutputSize	auto
LossFunction	crossentropyex

Figure 10: Clasification layer


fullyConnectedLayer ?

Name	<input type="text" value="fc"/>
InputSize	auto
OutputSize	<input type="text" value="10"/>
Weights	<input type="text" value="[]"/>
Bias	<input type="text" value="[]"/>
WeightLearnRateFactor	<input type="text" value="10"/>
WeightL2Factor	<input type="text" value="1"/>
BiasLearnRateFactor	<input type="text" value="10"/>
BiasL2Factor	<input type="text" value="0"/>
WeightsInitializer	<input type="text" value="glorot"/> ▼
BiasInitializer	<input type="text" value="zeros"/> ▼

Figure 11: Full connected layer parameter

4.3 CNN with pre-trained ResNet50

In this architecture, we have used pre-trained Resnet50 the same way as imageNet with CNN architecture. We have replaced the last three layers for transfer retraining and Connected the last transfer layer to new layers.

5 Results and Model evaluation

The dataset has 1000 pictures of 10 categories in total. We divide the training set and validation data set according to the ratio of 8:2.

From Table 2 and Figures 12, 13, and 14, we see that the basic CNN model has performed lowest compared to the pre-trained model CNN with imageNet and resNet50. It has only 40% train accuracy and 38.5% validation accuracy. Otherhand, both imageNet and ResNet50 have performed similar 78.5% and 79% validation accuracy. Though we see a little 0.5% outperformance for resNet50 compared to imageNet, however, we observed little overfitting issues for the resNet50 model compared to imageNet.

Model Name	Train Accuracy	Validation Accuracy
CNN	40.5	38.5
CNN with ImageNet	82	78.5
CNN with Resnet50	94	79.5

Table 2: Results of different models CNN, ImageNet, and ResNet50.

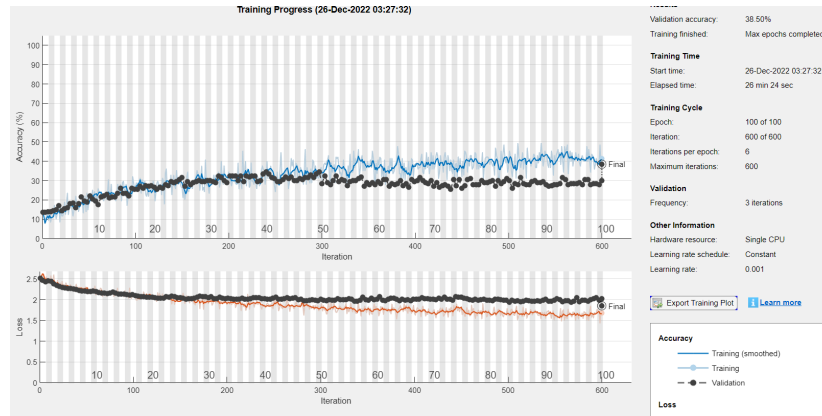


Figure 12: Basic CNN model training accuracy 40%, validation accuracy 38.5%.

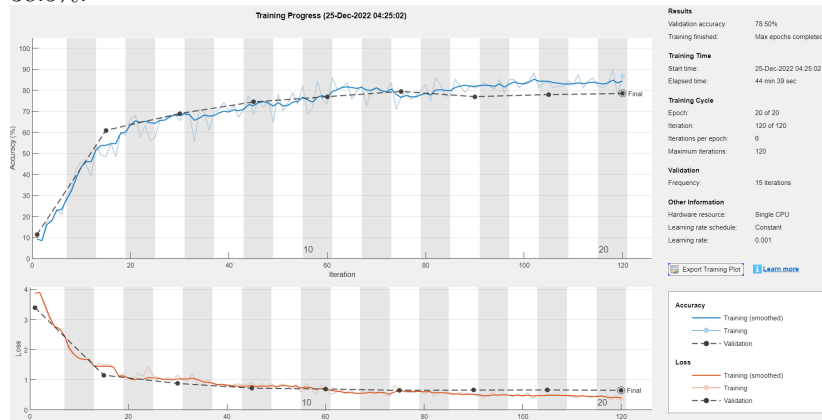


Figure 13: CNN model with pre/trained imageNet training accuracy 82%, validation accuracy 78.5%.

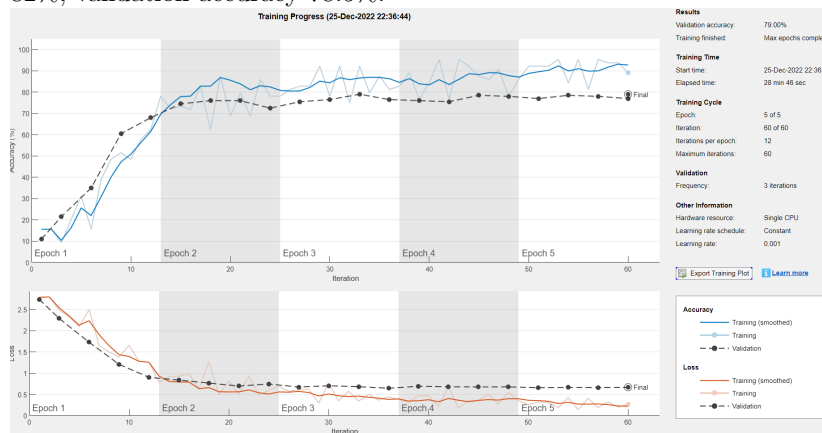


Figure 14: CNN model with pre/trained resnet50 training accuracy 94%, validation accuracy 79%.



Figure 15: Output classification random 4 samples.

Confusion Matrix												
Output Class	fabric	18 9.0%	0 0.0%	0 0.0%	1 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.7% 5.3%	
	foliage	0 0.0%	20 10.0%	0 0.0%	0 0.0%	0 0.0%	3 1.5%	0 0.0%	0 0.0%	1 0.5%	83.3% 16.7%	
	glass	0 0.0%	0 0.0%	17 8.5%	0 0.0%	0 0.0%	1 0.5%	2 1.0%	1 0.5%	1 0.5%	77.3% 22.7%	
	leather	0 0.0%	0 0.0%	0 0.0%	14 7.0%	1 0.5%	1 0.5%	0 0.0%	0 0.0%	0 0.0%	87.5% 12.5%	
	metal	0 0.0%	0 0.0%	1 0.5%	2 1.0%	16 8.0%	0 0.0%	3 1.5%	0 0.0%	0 0.0%	64.0% 36.0%	
	paper	2 1.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	11 5.5%	4 2.0%	0 0.0%	0 0.0%	64.7% 35.3%	
	plastic	0 0.0%	0 0.0%	1 0.5%	0 0.0%	0 0.0%	3 1.5%	11 5.5%	0 0.0%	0 0.0%	73.3% 26.7%	
	stone	0 0.0%	0 0.0%	0 0.0%	2 1.0%	1 0.5%	0 0.0%	0 0.0%	16 8.0%	0 0.0%	80.0% 20.0%	
	water	0 0.0%	0 0.0%	1 0.5%	0 0.0%	1 0.5%	0 0.0%	0 0.0%	1 0.5%	18 9.0%	85.7% 14.3%	
	wood	0 0.0%	0 0.0%	0 0.0%	1 0.5%	1 0.5%	1 0.5%	0 0.0%	2 1.0%	0 0.0%	16 8.0%	
		90.0% 10.0%	100% 0.0%	85.0% 15.0%	70.0% 30.0%	80.0% 20.0%	55.0% 45.0%	55.0% 45.0%	80.0% 20.0%	90.0% 10.0%	80.0% 20.0%	78.5% 21.5%
		fabric	foliage	glass	leather	metal	paper	plastic	stone	water	wood	
Target Class												

Figure 16: Confusion matrix of CNN with imageNet

From the confusion matrix, we can see that some classes performed very well; for example, Fabric, foliage, and water classification performed with 90 to 100% of accuracy. However, some classes, for example, paper and plastic, has only 55% of classification accuracy. If we closely observe, our classifier has confusion in these two classes mostly; for example, a good number of Paper classes are misclassified as Plastic classes, and similarly, a good number of Plastic classes are misclassified as Paper as well.

6 Conclusion

In this project work we classify the material data of 10 classes by using Convolution Neural Networks (CNN) as a baseline and finally compare with a more advanced approach like CNN with pre-trained model GoogleNet and ResNet-50. We did different preprocessing, and among flipping and rotation did improve classifier performance; however, image grayscale conversion did not help. Among the model selection, we found CNN without any pre-trained performed worse; however, imageNet and resNet50 performed very well with 79% of accuracy. We did observe little overfitting issues, and especially two classes (paper

and plastic) were confused with each other. After the classification result, model learning, and confusion matrix analysis, we can strongly say that our model is performing well despite having a small dataset. Since we have 10 classes and only have 1000 samples total for training and validation, that means we had only 80 samples for each class to train the model. We believe if we have more data samples, then the minor overfitting issues and less performance of paper and plastic classes could be solved.

References

- [1] L. Sharan, R. Rosenholtz, and E. H. Adelson, “Material perception: What can you see in a brief glance?,” *Journal of Vision*, vol. 9, pp. 784–784, 2010.
- [2] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, “Adaptive data augmentation for image classification,” in *2016 IEEE international conference on image processing (ICIP)*, pp. 3688–3692, Ieee, 2016.
- [3] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.