Texturoid: A micro-macro texture descriptor

Mayank Mohindra
mayank15056@iiitd.ac.in
Indraprastha Institute of Information Technology
New Delhi, Delhi, India

Pulkit Goel
pulkit15158@iiitd.ac.in
Indraprastha Institute of Information Technology
New Delhi, Delhi, India

ABSTRACT

In this paper, we review the modern methods and datasets available specifically for texture classification, and evaluate the state of the art LBP-Scattering Transform based Texture Classifier. In this paper Nguyen et al.[8] have mentioned a novel technique to describe textures in image and classify them. This paper aims to propose some limitations of the approach mentioned and also further their research using existing popular techniques for texture classification Code and Dataset: **Link to source code Github Repo**

KEYWORDS

Local binary pattern, deep learning, performance evaluation, texture classification, image classification, image texture analysis, Wavelet Transforms, Scattering Transforms

1 PROBLEM STATEMENT

To determine salient features in texture images and work towards correct classification of textures.

2 INTRODUCTION

To understand the purpose of the research one must understand what textures are and how are the classification based approaches different from standard image analysis approaches meant for objects found in the images and their enhancement.

According to Pietikäinen et al.[10], textured image can be characterized by a non-uniform or varying spatial distribution of intensity or color. These textures posses some unique features like the classification capability using only a small fragment of a texture. Approaches specifically meant for object classification, involving the use of pixel intensities and their correlation, could also be directly applied to textures but they fail to take advantage of the uniqueness that the textures. Also certain object classifiers prioritize shapes that the image contain over the way the pixels are distributed. Texture classifiers thus analyze the pixel structures and the patterns that they enumerate. Thus there is an ardent need for strong texture classifiers.

Texture classification has found its application in many varied domains with conclusive evidence. It has found its way to computer-aided medical diagnosis described by Harms et al.[5], classification of forest species performed by Tou et al.[11], geo-processing by Haralick et al.[3] and agriculture demonstrated by Jhuria et al.[6], and Mohana et al.[7]

Many methods to classify textures have been proposed through the years and they have evolved over time. All these approaches may be subdivided into three categories: the texture descriptors, describing the textures using well defined metrics and providing us with the feature set from the textured images to be used with standard classifiers; patch-based methods, which exploit the repeating nature of the patterns by increasing the training and testing datasets after dividing each texture image into sub-sections and thus using them to train the model, and test against individual parts of the test texture image thus classifying to the one with highest occurrence; and modern methods employing deep neural networks and CNNs.

The mentioned approaches have their individual pros and cons and we aim to exploit their nature into finding a combination that tries to best classify the textures into their classes as proposed by the state of the art Nguyen et al.[8], giving us with the classifier as a combination of the textual descriptor and neural learning.

The paper is divided into five sections. Section 3 is the literature review focussing on the existing approaches highlighting their abilities and limitations; Section 4 aims to describe in brief, the attempt by Nguyen et al.[8] in classifying textures to the best of their ability which may also be considered as our reference point for our attempt to propose advancements of the same; Section 5 contains information of the texture datasets and links that will be considered and tested upon; Section 6 briefly highlights our present plans to further the approaches already mentioned and work towards better texture classification performing our analysis on the aforementioned datasets; Section 7 explains our tentative time-line as to how we plan to achieve these results; Section 8 concludes the paper.

3 LITERATURE REVIEW

Texture classification has gone through a rough road with many available publications and researches to refer to. But on a broad scale, they all may be categorized into three categories. In this section we would define the three categories and also mention in brief what they are about, to further our understanding in them and to give explanation why we are using these techniques in our proposed solution.

3.1 Texture Descriptors

Many time one might be stuck in difficult situations when datasets available are small and not worth training the neural network else they would essentially make the model produce inaccurate results. Such cases make the texture descriptors imperative. Also some neural learning approaches also point out a broader perspective while the dataset's might require understanding of the fine variations. For this the features in texture must be defined. This section would analyze some of the popular existing approaches to texture description. This will also form the basis of the advancements we plan to perform in the local variations section of Phase III in our Section IV.

Conference'17, July 2017, Washington, DC, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.1.1 Gabor Filters Banks. The Gabor filter uses frequency and orientation as their descriptors in defining the textures. These are represented by complex and bidirectional sine functions. These in combination with eccentricity and symmetry form what we call the Gabor filter Banks. The problem with Gabor is that it allows for varying a high set of parameters, and it is difficult to find the optimal values for them. This was also pointed out by Bianconi et al.[2]

3.1.2 Local Phase Quantization (LPQ). This approach uses a 2D fourier transform and using the same, the local phase information is extracted. This phase information can be used to describe the texture. This is also called the short term fourier transform.

3.1.3 Gray-Level Co-occurrence Matrices (GLCM). This approach defines three parameters which are Energy, Contrast and Entropy. These definitions are normalised functions of distance D and angel θ. This is one of the promising apporaches and we plan to test this against the implementation of LBP as described in Section 3.

3.1.4 Local Binary Patterns (LBP). LBP, as explained by Ojala et al.[9], works by encoding the pixels with some values and then representing the texture images with these codes. Later Guo et al.[4] suggested an advancement to LBP called Complete LBP (CLBP) which decomposes the image differences two operators referring to Sign and Magnitude of the encoders. Later using these operators the encoding is performed and the LBP histogram is computed. This approach is also the one used by authors Nguyen et al.[8] as described in Section 3.

3.2 Patch-Based Classifiers

The main difference between regular images encompassing objects and textured images incorporating only texture is that the latter is repeating in nature. The patch-based classifier is used when we wish to increase the training set or incorporate fine grained differences in the texture and thus improve classification. The entire textured image is divided into what we call patches (split into small segments) and then the training data is recomputed. The feature extraction and description phase may follow.

The recompilation of the segments can be done either:

- after the feature selection procedure in which case we can leave the test case intact.
- after the classification step in which case the test data must also be patched and then the weighing texture is assigned to the compiled test case.

3.3 Deep learning approaches

With the advancements in machine learning, and specifically deep learning and CNNs, the field of texture description and classification has attracted the attention of many researches using them to provide reliable predictions and classifications. The paper uses scattering transform and has been discussed thoroughly below. Apart from that there are certain other architectures specifically designed for texture classification. One such example is the TCNN proposed by Andrearczyk et al.[1] This approach is also discussed in Section 4 and we plan to change the wavelet CNN used by the authors in Section 3 by TCNN implementation.

3.4 Dimensionality Reduction

3.4.1 Principal Component Analysis. As the name suggests, PCA is used for determining which features best describe the dataset and which can be used by classifiers to classify the test images. This is also called dimensionality reduction. We will not go into the mathematical details. In simple steps we can summarise PCA as follows:

- Perform singular value decomposition of the obtained covariance matrix
- Find the eigenvector and eigenvalue pairs after the decomposition.
- Sort these pairs on the basis of eigenvalues and store the corresponding eigenvectors.
- These eigenvectors are then used to reduce the dimensionality.

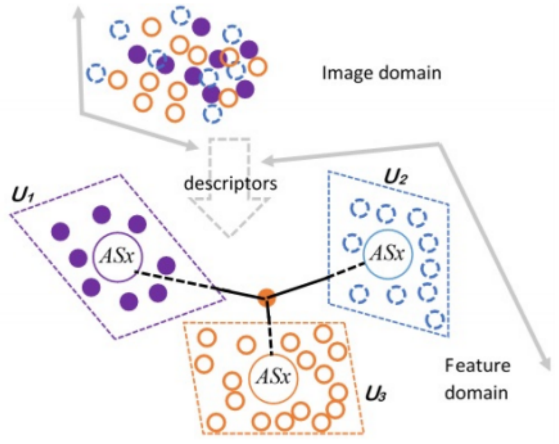


Figure 1: PCA-classifier classifies a test image X based on the minimum distance from Scattering Transform

3.4.2 Linear Discriminant Analysis. Just like PCA, LDA is another example of feature reduction and selection. The basic difference is that LDA utilises the concept of calsses and rather than maximizing the variance in the data, LDA aims to find the direction that maximizes the inter class distance, thus find aiming to find a clear separation between data points based on classes. This thus required labelled data as input.

3.4.3 Hybrid. PCA and LDA are often used one after the other (PCA over LDA or LDA over PCA) to perform analysis. We plan to use this approach in Phase III of Section 4.

3.5 Scattering Transform

Although this is a neural learning approach, it deserves a special mention since this is actively used in Section 3 by the authors.

A scattering transform is used to build a stable,invariant and informative signal representations of any type of classification. The most basic way to calculate it is by computing the signal information across multiple paths and use a convolutional network to cascade wavelet modulus operators. It is used for texture,audio as well as image discrimination since it is quite stable to deformations.

A scattering network is used to calculate the actual scattering transform. Each of the layer of deep convolutional network is calculated by consecutively calculating wavelet transform and modulus operator. Each wavelet transform outputs two things : First a scattering coefficient for the same layer and second a wavelet modulus coefficient for the next layer. The second coefficient is further transformed to a wavelet transform. All the low frequencies

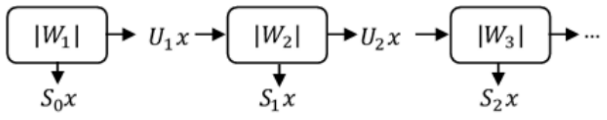


Figure 2: Scattering Transform

are carried by the average scattering transform which loses the high frequencies in each step.The roto-translational convolution wavelets captures the high frequencies releasing the low frequencies in subsequent steps.

Calculating the wavelet modulus coefficient for each layer helps in generating the scattering coefficient for each layer. The final scattering coefficient is the concatenation of all the coefficients of all the layers which is finally used to calculate the transform.

3.6 Biologically Inspired Filtering (BF)

As the name suggests, this is a filtering technique inspired by the mechanism the human eye uses to extract detailed information from the images analyzed. This is generally used as a preprocessing step and is successfully used in Section 4 by the authors. In this technique, the image is first filtered using band pass filters and then decomposed into two maps corresponding to the image before the feature extraction step. The mathematical complexities are beyond the scope of the paper.

4 STATE-OF-THE ART DESIGN

For the purpose of reference, we use the novel approach proposed by Nguyen et al.[8] in the paper titled A Scattering Transform combination with Local Binary Pattern for Texture Classification. As the name suggests, the paper uses a modified version of LBP and scattering transform for describing texture images and thus facilitate texture classification.

The entire approach can be divided into two steps:

4.1 Part 1

Micro structure feature extractor

When we say micro, we refer to the features local to regions (like the local minima) and in texture based methods, dependent on the datasets, some might regularly require to capture the smaller variations of the image. To do this, the authors have proposed a two step approach

4.1.1 Step a. **BF**

This step involves the use of Biologically Inspired Filtering (BF). BF divides the image into two decomposed sets and then passes to the CLBP to work upon. More information is in the Literature Review.

4.1.2 Step b. **CLBP**

This step takes input from BF as two decomposed sets and then perform the CLBP encoding scheme. According to the steps of the approach we get a CLBP encoding matrix after the recomposition of the two steps. After this step we have accounted for the local variations.

4.2 Part 2

Macro structure feature extractor

The macro structure feature extractor refers to the use of the scatter transform step. Macro is a reference to the the global feature extraction in the texture image. For more details on how the scatter transform works, refer to the literature review. This step is referred to as the ScatNet by the authors. Scattering representation is computed by a cascade of wavelet modulus operators $|W_m|$.

Every $|W_m|$ has one input and two outputs which are the scattering coefficients S_{mx} and the next layer wavelet modulus coefficients $U_{(m+1)}$x. The latter is used for further transformation.

After this the features from both Parts are combined and the final features are a concatenation of both parts.

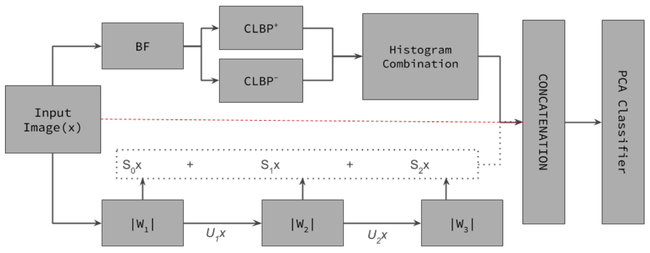


Figure 3: Algorithmic Framework

5 PROPOSED SOLUTION

Building on to the works of Nguyen et al.[8], we aim strengthen the proposed algorithm to the best of our ability. For the purpose of explaination of our (still young) approach, we divide the algorithm steps in four phases:

5.1 Phase I: The Input Phase

This is the phase in which we pass the input texture images to the Hybrid Descriptor (Phase II). We notice that the preprocessing if any is done in the descriptor and not much preprocessing is considered. To we plan to draw the reader's attention to the previously mentioned approach Patch-Based approaches to texture classification.

Since the images are texture images we plan to exploit their repeating nature by splitting the images into multiple well defined patches which are combined after at the feature extraction step (Phase III). This would essentially add weight to the training and test data and will incorporate the local variations in the textures carefully.

One might argue that this step would be similar to Biologically Inspired Filtering, used in the descriptor, but this is different since the thresholding of the local regions is user controlled and one might easily tune parameters to determine the level of the locality which we want to specify; also Biologically Inspired Filtering uses Difference of Gaussians approach thus working on the continuous range while the when preceded by the patches, its effect on smaller locality will essentially enable in capturing fine grain variabilities should they be tuned to do so, which we believe will find many applications especially in the medical industry analysing the tissue textures.

5.2 Phase II: Hybrid Descriptor

This phase as explained by the authors uses a hybrid of Biologically Inspired Filtering(BF) + Complete Local Binary Pattern (CBLP) and ScatNet, a scattering transform based neural learning implementation. The hybrid descriptor revolves in two steps:

5.2.1 Local texture variation capture. This step involves using BF on the input supplied by Phase I and follow it with CLBP, which as explained earlier are well used popular examples of texture descriptors. We plan to propose the introduction of Histogram of Gradients in combination to further their approach. HOG, as reported by Xiaoyu Wang et al.[13] is known to further improve the accuracies when used with LBP. We plan to test the same postulates when combined with CLBP and compare results. Another approach that we plan to implement is the use of SIFT transform, which just like HOG is another feature descriptor. SIFT is partially invariant to affine distortion, this may be used to disregard small changes in the geometry of the object and focus mainly on the texture.

5.2.2 Global texture variation capture. For this the authors have suggested the use of Scattering transform based neural learning approach. This as stated by the authors is a macro detection based method used for capturing the essence of the texture image in totality and not the input segment. Since its purpose is to capture variations at the image level we must redirect its input directly from the input images thus bypassing the patching step. The phase is said to have 3 layers, all of which performing wavelet transformations (2D or 3D). However this step is elaborate in itself, we plan to further the research by replacing scatNet with suggestions by Andrearczyk et al.[1] in 2016 in their paper on the implementation of Texture CNN (TCNN) which is a neural learning approach specifically implemented for texture based classification.

5.3 Phase III: Feature selection

After Phase II, all the features that we have will be well described and ready to be acted upon. This means that now the phase involved is the step of reducing the dimensionality and extracting the features that actually contribute towards the classification phase (Phase IV). The authors have suggested the use of Principal Component Analysis (PCA) as the prime feature selection method. We are aware of another well defined and popular approach of Linear Discriminant Analysis (LDA). We plan to use a well defined hybrid of either PCA over LDA or vice versa to improvements in the feature selection phase.

5.4 Phase IV: Classification

This phase has not been defined by the authors since the PCA in Phase III is also capable of classifying the results and this is what was done. We would like to test against the following classifiers:

- LogisticRegression
- SVC
- KNeighborsClassifier
- MLPClassifier
- KNeighborsClassifier
- GradientBoostingClassifier
- RBF
- RandomForestClassifier
- GaussianNB

This phase has not been reported by the authors and thus is very promising and could potentially provide the necessary boost in accuracy. So after updating all the phases below is our final proposed new architecture for texture classification problem.

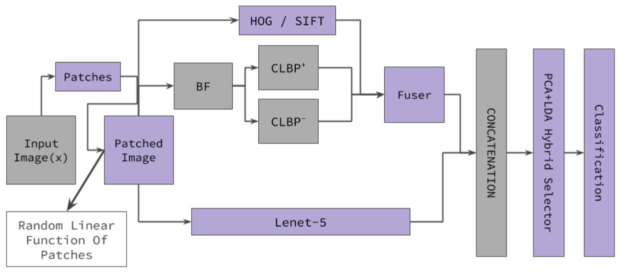
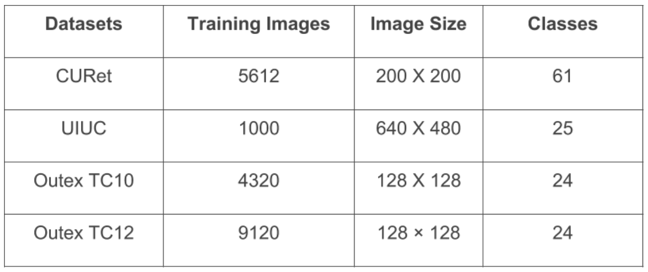


Figure 4: Our final proposed architecture

6 DATASETS



Datasets	Training Images	Image Size	Classes
CURet	5612	200 X 200	61
UIUC	1000	640 X 480	25
Outex TC10	4320	128 X 128	24
Outex TC12	9120	128 × 128	24

Figure 5: Dataset Analysis

6.1 DataSet 1

Amsterdam Library of Textures (ALOT) It has a color image collection of 250 rough textures(classes) . Each material(class) has 100 samples. There are additional 12 textures(classes) which makes a total count of 27500 samples.

6.2 DataSet 2

Columbia-Utrecht Reflectance and Texture Database (Curet) It has about 61 texture classes. There are 205 samples for each class which are captured at different angles and orientations. Finally only 92 images were chosen for each class with 61 classes finally which makes 5612 total samples.

6.3 DataSet 3

Outex Texture Database Only two types of suites Outex TC 00010 (TC10) and Outex TC 00012 (TC12) were used for training and testing purposes. There are about 24 classes and each class has around 200 sample. All the images are captured under three types of illuminations.

6.4 DataSet 4

UCIC database has 25 different types of textures(classes) . Each class has around 40 samples. Finally 20 samples were chosen for training and testing and the average accuracy was calculated by a random split of 10

6.5 DataSet 5

The KTH-TIPS and KTH-TIPS2 image Database This has 2 types of suites a and b. Each suite has around 9 classes and there were 432 samples per texture (class). Each image was captured using different angle and illumination.

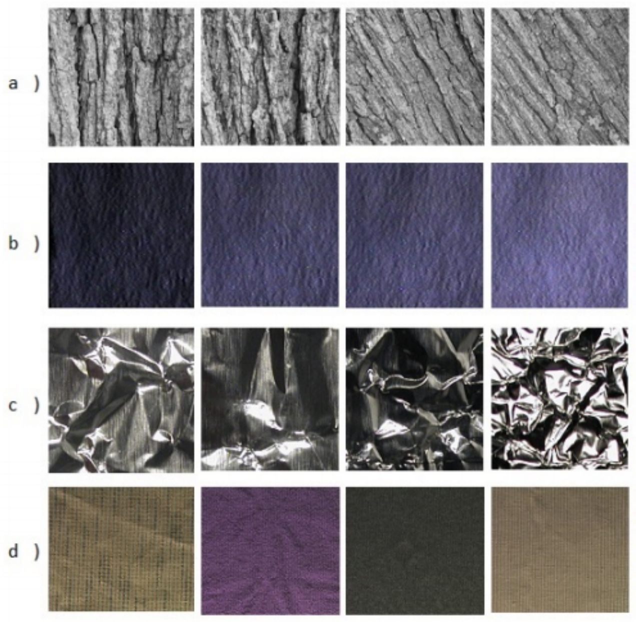


Figure 6: Images in the same class from a) UIUC, b) CUReT, c) KTH TIPS2b, d) Outex datasets

7 INTERMEDIATE PROGRESS REPORT

7.1 Datasets Used

For the initial implementation we thought of implementing out approaches on small subset of datasets and later implement on complete dataset. To test our initial approach and for base line results we used subset of Outex and UIUC datasets.

(1) Outex Dataset:- The subset used was Outex_TC_00010 from Outex dataset. It consists of 4320 training images and 723 testing images. It consists of 24 different classes of textures.
(2) UIUC Dataset :- The subset of this dataset consisted of 1000 images which were splitted training and testing sets in ratio of 4:1.

7.2 Techniques Implemented

(1) ScatNet: The first step in computing scattering coefficients is to perform wavelet transformation on each image. After performing wavelet transformation scatnet function is used to compute the coefficients. The source code of ScatNet was available in Matlab. However we translated the code into python for this project. The following images shows the second order scattering with 5 scales and 6 orientation per layer.

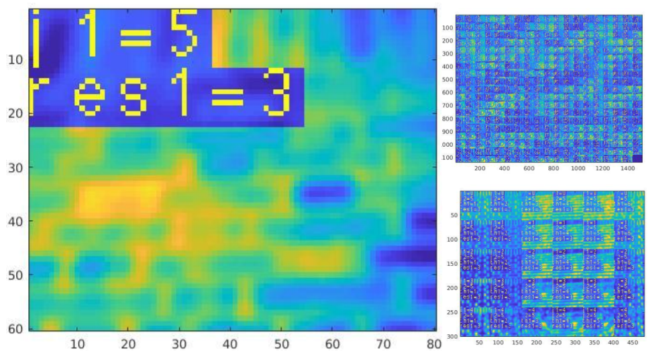


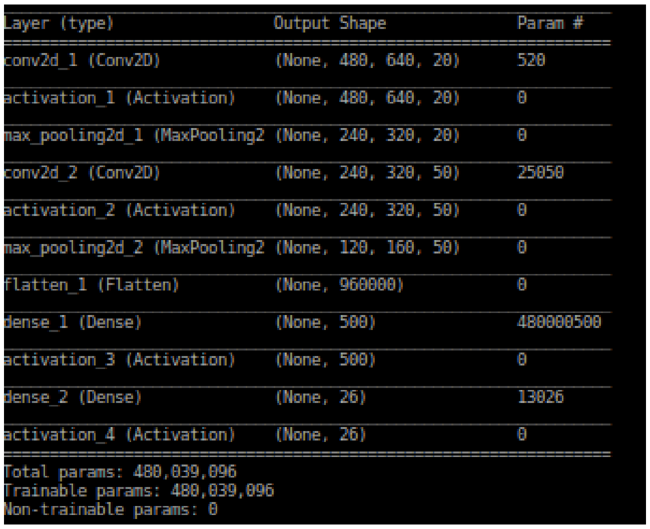
Figure 7: Second order scattering with 5 scales and 6 orientations per layer

(2) CLBP: The source code for CLBP was available. We modified the code for our problem statement and incorporated the Outex and UIUC datasets. In CLBP the image local regions are decomposed into two complementary components. The CLBP-Sign components is same the LBP and the CLBP-Magnitude component is local variance of the magnitude.
(3) BF: The source code for the biologically inspired filtering was unavailable so we implemented using the approach mentioned in [12]. The approach is as follows:
    (a) pass the images through a band pass Difference of Gaussians (DoG) and just like the bipolar cells in the human eye, it mimics its functionality.
    (b) Once we obtain the responses of the bipolar, we decompose them into two maps, each corresponding to the image details alongside the two sides of the image edge.

(c) Then the ideation of performing feaure extraction and description followed in the implementation phases are performed individually on the two obtained maps spearately. These maps can be called the positive and negative components of BF obtained.

(4) PCA and LDA: As mentioned before, the authors have used the process of Principal Component Analysis (PCA), for both feature reductions and classification. We had proposed an amalgamation of Linear Discriminant Analysis (LDA) and PCA or vice-versa. This is one of the many famous techniques used for feature reduction. The classification step is followed in the next subsection. Using PCA before LDA regularizes the problem and also avoids over-fitting, thus possibly giving better accuracy across multiple datasets. We cannot be sure of this since the integration with the final phases is still incomplete. But we have the amalgamated feature reducer to extract the most informative features and thus help increase accuracy and imporve performance. This filter is again implemented using Python's inbuild packages.

(5) LeNet and ResNet18 : To explore the deep learning side we implemented Lenet using Pytorch framework for Deep learning in python and Resnet using Keras framework. After many experiments we found out that Resnet performed better than Lenet so we used that in our final architecture. Model summary is given below for both the architecture.

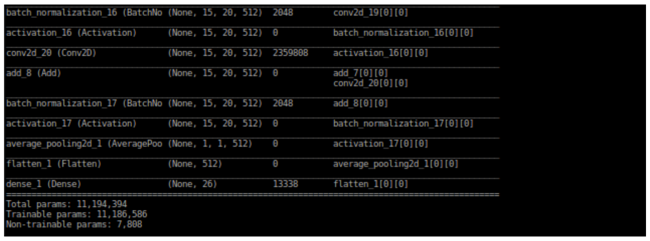


```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 480, 640, 20)      520
_________________________________________________________________
activation_1 (Activation)    (None, 480, 640, 20)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 240, 320, 20)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 240, 320, 50)      25050
_________________________________________________________________
activation_2 (Activation)    (None, 240, 320, 50)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 120, 160, 50)      0
_________________________________________________________________
flatten_1 (Flatten)          (None, 960000)            0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               480000500
_________________________________________________________________
activation_3 (Activation)    (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 26)                13026
_________________________________________________________________
activation_4 (Activation)    (None, 26)                0
=================================================================
Total params: 480,039,096
Trainable params: 480,039,096
Non-trainable params: 0
```

Figure 8: Summary of LeNet Model

7.3 Classification

The classification steps as mentioned in Section 5.4, is one of the most promising steps, since the authors have used PCA classifier only. Section 5.4 also mentions the possible use and implementation of many classifiers. Since we would be using the Python programming environment, we would easily be able to find the packages necessary to implement these classifiers. The only requirement left would be to tweak the input for theses classifiers to act upon.



```
batch_normalization_16 (BatchNo (None, 15, 20, 512)  2048        conv2d_19[0][0]
__________________________________________________________________________________
activation_16 (Activation)      (None, 15, 20, 512)  0           batch_normalization_16[0][0]
__________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 15, 20, 512)  2359808     activation_16[0][0]
__________________________________________________________________________________
add_8 (Add)                     (None, 15, 20, 512)  0           add_7[0][0]
                                                                 conv2d_20[0][0]
__________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 15, 20, 512)  2048        add_8[0][0]
__________________________________________________________________________________
activation_17 (Activation)      (None, 15, 20, 512)  0           batch_normalization_17[0][0]
__________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, 1, 1, 512)    0           activation_17[0][0]
__________________________________________________________________________________
flatten_1 (Flatten)             (None, 512)          0           average_pooling2d_1[0][0]
__________________________________________________________________________________
dense_1 (Dense)                 (None, 26)           13338       flatten_1[0][0]
==================================================================================
Total params: 11,194,394
Trainable params: 11,186,586
Non-trainable params: 7,808
```

Figure 9: Summary of ResNet18 Model

We have successfully implemented PCA classifier for CLBP implementation and tested for the OUTEX dataset for the reported accuracies. PCA performs the dual function of feature extractions and classification in this case.

Now for the ScatNet phase of progress, we have used ScatNet for extracting features and now we have used the SVM classifier on top for classification. This step is tested using the UIUC dataset and the accuracies are reported.

7.4 Results

So as discussed above we first implemented all phases individually and later combined all the features using feature fusion approaches. We started with local features and implemented HOG along with CLBP on UIUC dataset. This approach along with SVM classifer gave us an accuracy of 45% alone. After this we stared experiments with global features. In global features we implemented state of the art Deep learning architectures such as ScatNet, Lenet-5 and ResNet. Out of all the three Resnet gave the best accuracy of 93.8% on UIUC dataset. Now we combined these global and local features to further increase the accuracy.

(1) **Feature Fusion:** Our main aim was to combine the individual implementations of HOG and CLBP with global features like CNN(ResNet) and then perform analysis on the fused and extracted features thus giving us the required implementation. So with SVM as classifier and with combined features from HOG, CLBP and ResNet we got a final accuracy of **96.8%** on UIUC dataset.

Code and Dataset:
$https : //github.com/mayankmtg/Texturoid_{t}exture_{c}lassifier.git$

8 CONCLUSION

Texture classification is a salient problem and has received a lot of attention over the years. The approach suggested by the authors Nguyen et al. [8] is novel in the sense that it uses the combination of multiple approaches en an elegant amalgamation. But we felt that there are many other techniques that the authors have failed to address. Therefore we proposed a new architecture that use some of the state of the art techniques to further increase the accuracy.

REFERENCES

[1] Vincent Andrearczyk and Paul F Whelan. 2016. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters* 84 (2016), 63–69.

[2] Francesco Bianconi and Antonio Fernández. 2007. Evaluation of the effects of Gabor filter parameters on texture classification. *Pattern Recognition* 40, 12 (2007), 3325–3335.

[3] Huawu Deng and David A Clausi. 2004. Gaussian MRF rotation-invariant features for image classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2004), 951–955.

[4] Zhenhua Guo, Lei Zhang, and David Zhang. 2010. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing* 19, 6 (2010), 1657–1663.

[5] Harry Harms, U Gunzer, and Hans M Aus. 1986. Combined local color and texture analysis of stained cells. *Computer vision, graphics, and image processing* 33, 3 (1986), 364–376.

[6] Monika Jhuria, Ashwani Kumar, and Rushikesh Borse. 2013. Image processing for smart farming: Detection of disease and fruit grading. In *Image Information Processing (ICIIP), 2013 IEEE Second International Conference on*. IEEE, 521–526.

[7] SH Mohana and CJ Prabhakar. 2015. A novel technique for grading of dates using shape and texture features. *arXiv preprint arXiv:1501.01090* (2015).

[8] Vu-Lam Nguyen, Ngoc-Son Vu, and Philippe-Henri Gosselin. 2016. A scattering transform combination with local binary pattern for texture classification. In *International Workshop on Content-based Multimedia Indexing*.

[9] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence* 24, 7 (2002), 971–987.

[10] Matti Pietikäinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. 2011. *Computer vision using local binary patterns*. Vol. 40. Springer Science & Business Media.

[11] Jing Yi Tou, Phooi Yee Lau, and Yong Haur Tay. 2007. Computer vision-based wood recognition system. In *Proceedings of International workshop on advanced image technology*. Citeseer.

[12] Ngoc-Son Vu, Thanh Phuong Nguyen, and Christophe Garcia. 2014. Improving texture categorization with biologically-inspired filtering. *Image and Vision Computing* 32, 6-7 (2014), 424–436.

[13] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. 2009. An HOG-LBP human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 32–39.