Histogram Layer for Texture Classification

Tyler (Taewook) Kim
Kent School
1 Macedonia Rd., Kent, CT 06757
kimt20@kent-school.edu

Joshua Peeples
University of Florida
1064 Center Dr, Gainesville, FL 32611
jpeeples@ufl.edu

Abstract

In this paper, we propose a hybrid model that incorporates a stackable, localized histogram layer on convolutional neural network (CNN) for texture analysis applications. Texture classification is a part of texture analysis that focuses on the image's spatial variation in pixel intensity values and gray levels to classify texture. Instead of using standard histogram operation, we used RBF (Radial Basis Function) to perform a localized binning operation without binning constraints. In order to evaluate the proposed model's performance, we used six different CNNs comprising of histogram and convolution layer to classify images. Experiments were conducted using the KTH-TIPS 2b dataset and the use of standardization was analyzed during this study. As a result, every network improved performance significantly with standardization, averaging about 6.50% increase in average overall accuracy. Then, the results were evaluated using confusion matrices, classification metrics, and overall accuracy on the networks standardized data to further evaluate the proposed model. Overall, the proposed model did not necessarily perform better than the standard CNNs, but the histogram CNN performed better on some classes. Future works will include evaluating different initialization methods and parameters such as number of bins, window size, and kernel size.

1. Introduction

Texture analysis is a process of characterizing texture in an image by spatial variation in pixel intensity values or gray levels. Image texture is a complex visual pattern that is one of the crucial sources of visual information. Some of the texture properties include but not limited to perceived lightness, uniformity, density, roughness, and granulation [6]. This field has been one of the popular research topics in computer vision and machine learning due to its broad impact on numerous fields [2]. In medical imaging, texture analysis can quantify the tumor heterogeneity based on MRI, CT, or PET images [7]. In [12], researchers implement texture analysis based on leaf images to classify various crop diseases.

Approaches to texture analysis are generally categorised into four approaches: structural, statistical, model-based, and transform. **Structural analysis** represent texture by microtexture and macrotexture based on pre-defined primitives and the placement rules. **Statistical approaches** represent texture indirectly with methods such as analyzing statistics given by pairs of pixels. **Model-based texture analysis** interprets an image texture using fractal and stochastic models. Lastly, the **transform methods** of texture analysis use methods such as Fourier, Gabor, and Wavelet transforms to interpret texture characteristics such as frequency or size [6].

1.1. Related Works

Texture analysis is similar to object recognition but it differs because spatial relationships of patterns in the image are important for most texture analysis approaches, as opposed to points of interests for object recognition [2]. Texture-based datasets also have higher dimensionality compared to simple color and shape-based datasets used in object recognition, which makes it a harder task [1]. In order to confront the complexity of texture datasets, several different methods to perform texture analysis have been proposed throughout the literature.

One of the previous approaches is a traditional approach where researchers used different texture descriptors to observe the region homogeneity and the histograms of these region borders. In [1], the authors present a review of various hand-crafted features including traditional texture descriptors such as Local Binary Patterns (LBP) [11] and Gray-Level Co-Occurrence Matrices (GLCM) [3] as well as a patch- and multiscale-based approaches. These methods have been used for the past decades and have proved their effectiveness in various applications.

With the recent advances on CNN, researchers have ap-

plied deep neural networks to improve performance and avoid the laborious process of developing hand-crafted features. Rather than manually designing features for the machine learning model, deep learning approaches are capable of automatically extracting features from labeled datasets and yield higher performance. However, deep neural networks such as CNN require a copious amount of labeled data along with immense amounts of processing power.

In recent years, there have been several studies where researchers combined the deep learning approach and traditional hand-crafted features to create a hybrid model. Nguyen et al. [8] proposed a new presentation attack detection (PAD) method that uses CNN and the multi-level local binary pattern (MLBP) method together to form a hybrid feature with a higher discrimination ability. In [9], the authors proposed a hybrid model which can learn from both deep learning features derived from spectral data as well has hand-crafted features derived from synthetic aperture radar (SAR) imagery and elevation. Some of these hybrid models do not require an immense amount of labeled data but combine automated feature learning and traditional hand-crafted feature to improve performance [9]. Zhang et al., proposed a Deep Texture Encoding Network where they added an encoding layer on top of convolutional layers [16]. This model was able to learn convolutional features and encoding representation simultaneously. A theoretical analysis on implementing deep neural networks specifically for texture classification purposes was conducted to demonstrate the need for the integration of traditional and neural features [1]. These pioneering attempts showed improved performance compared to previous methods with deep learning algorithms or hand-crafted features alone.

There were also papers which focused on implementing histogram layers for various applications. In [13], the authors added a histogram layer to the CNN to mimic existing features such as projection spatial rich model (PSRM) and proposed the model for steganalysis applications. By implementing a new histogram layer, they were able to capture the statisctics from the feature maps with the histogram layer. Wang et al [14] proposed a learnable histogram layer that can backpropagate errors as well as learn the optimal bin centers and widths. Two architectures were developed for object detection and semantic segmentation, and both models were able to achieve high performance. However, previous studies were limited to using global histograms which caused loss of spatial information. They were also limited to using a single histogram layer.

1.2. Goal of Research

In this paper, we propose a novel model that incorporates a localized histogram layer for CNNs. Our studies will be the first attempt to implement such a hybrid model for texture analysis applications and implementing localized histograms will provide several advantages. 1) Spatial information, which is important for texture, will be retained as opposed to previous global methods. 2) Our approach will use radial basis functions (RBFs) which will relax the binning constraint because each feature value's contribution will be based on their proximity to each bin center and associated bin width. Additionally, this will enable the network to be less sensitive to various outliers and ambiguity in the data. 3) This new layer will use a differentiable histogram operation for deep learning algorithms as well. 4) Lastly, our model will have stackable histogram layers which is capable of capturing more higher-level features.

2. Methodology

2.1. Binning Operation

Binning operation in standard histogram can be expressed with the following counting fuction:

$$y_k = \begin{cases} 1, B_k - w \leq x_k < B_k + w \\ 0, otherwise \end{cases} \tag{1}$$

This indicator function returns 1 if the element falls into a bin and returns 0 else. The condition when an element falls into a bin is defined by $B_k - w \leq x_k < B_k + w$, B_k being the bin center and w being the bin width. However, the standard histogram operation is not differentiable and cannot be used for the backpropagation [1][14].

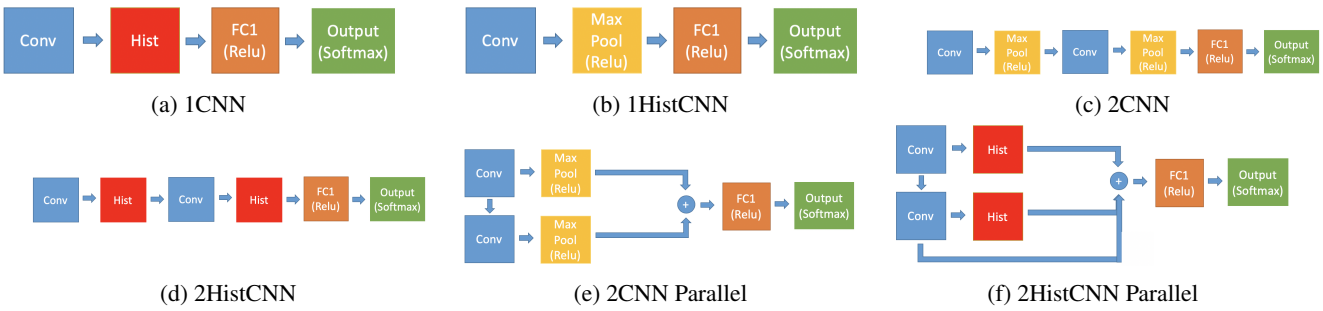
Therefore, our approach is to perform a localized binning operation with a sliding window. RBFs will be used instead of the standard histogram operation to approximate the count values:

$$y_k = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} e^{-\frac{(x_{ij} - \mu_k)^2}{\sigma_k^2}}. \tag{2}$$

The function above returns the count y_k, for kth bin center μ_k and width σ_k from the feature map value x_{ij} . For RBFs, μ_k value controls the location of where the function is; therefore, this parameter has similar functionality to that of a bin center location for histogram. A similar comparison can be made with w (bin width) and σ_k it controls the scale (spread) of the distribution.

An example with M×N window in an image is shown in Figure 2. Each square represents a pixel with a single intensity value (feature map value) of 1, 2, or 3. Every pixel will contribute to every bin but not equally due to the RBF we are using.

Table 1 compares the resulting count value from the two equations introduced. Each column represents the bins, b_0,



(a) 1CNN (b) 1HistCNN (c) 2CNN

(d) 2HistCNN (e) 2CNN Parallel (f) 2HistCNN Parallel

Figure 1: Architectures used for each texture dataset.

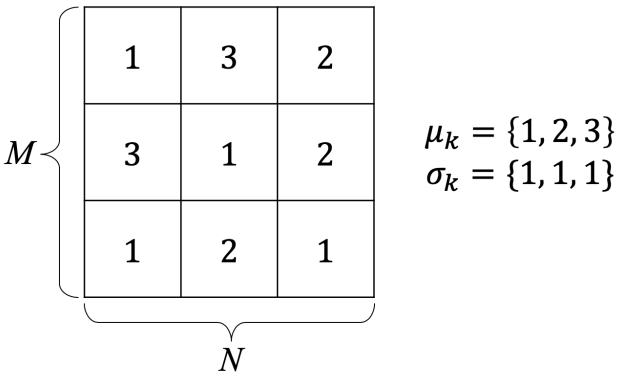


Figure 2: M×N Window

Table 1: Normalized Frequency Comparison

	b_0 **Value: 1**	b_1 **Value: 2**	b_2 **Value: 3**
Standard	0.44	0.33	0.22
RBF	0.57	0.42	0.35

b_1 and b_2 each indicating bin center of 1, 2, and 3 respectively. The first row shows the value calculated by Equation 1 with $w = 0$, and the second row is based on Equation 2. RBF is used to approximate the counting function because it allows differentiation thereby allowing the model to learn via backpropagation. As shown in Figure 2, σ_k is set to be 1, which makes the difference between the two values larger. σ_k is a crucial factor that determines this difference. As σ_k value becomes smaller, the model will only consider the values that are very close to the bin center. If model learns the optimum σ_k value, we will be able to account for ambiguity in the data as opposed to crisp histograms. As σ_k value becomes larger, every element will fall into the same bin.

Table 2: Number of Total Parameters for Each Network

1_CNN	**2_CNN**	**2_CNN_parallel**
10923	16583	65315
1_HistCNN	**2_HistCNN**	**2_HistCNN_parallel**
10931	16579	65435

2.2. Experiment Design

Figure 3: Example of KTH-TIPS 2b Images [5]

To evaluate the performance of the proposed model, we used KTH-TIPS 2b, a texture database with 4752 images classified into 11 categories with four samples in each class (A, B, C, D). We use six different artificial neural networks (ANN) that are comprised of convolutional and histogram layers to classify the images. We will compare the results both quantitatively and qualitatively. In qualitative analysis, confusion matrices are used to show the number of images that are classified correctly or incorrectly by each network. In quantitative analysis, we record the average accuracy, precision, recall, and F1 score.

Table 2 shows the number of parameters for each net-

Table 3: 1_CNN & 1_HistCNN

	1_CNN Avg. F1-score	**1_HistCNN Avg. F1-score**	**1_CNN Avg. Precision**	**1_HistCNN Avg. Precision**	**1_CNN Avg. Recall**	**1_HistCNN Avg. Recall**
Sample A	0.45 ± 0.24	0.31 ± 0.28	0.47 ± 0.22	0.30 ± 0.27	0.48 ± 0.29	0.36 ± 0.32
Sample B	0.43 ± 0.22	0.35 ± 0.29	0.43 ± 0.21	0.33 ± 0.30	0.47 ± 0.27	0.40 ± 0.32
Sample C	0.44 ± 0.25	0.28 ± 0.28	0.43 ± 0.24	0.29 ± 0.31	0.49 ± 0.28	0.33 ± 0.33
Sample D	0.40 ± 0.23	0.35 ± 0.30	0.42 ± 0.22	0.35 ± 0.29	0.46 ± 0.29	0.41 ± 0.32

Table 4: 2_CNN & 2_HistCNN

	2_CNN Avg. F1-score	**2_HistCNN Avg. F1-score**	**2_CNN Avg. Precision**	**2_HistCNN Avg. Precision**	**2_CNN Avg. Recall**	**2_HistCNN Avg. Recall**
Sample A	0.40 ± 0.23	0.33 ± 0.25	0.41 ± 0.22	0.31 ± 0.21	0.44 ± 0.28	0.41 ± 0.33
Sample B	0.40 ± 0.23	0.37 ± 0.24	0.40 ± 0.21	0.37 ± 0.24	0.44 ± 0.26	0.42 ± 0.31
Sample C	0.42 ± 0.22	0.34 ± 0.25	0.44 ± 0.22	0.34 ± 0.22	0.46 ± 0.29	0.40 ± 0.34
Sample D	0.38 ± 0.24	0.32 ± 0.24	0.40 ± 0.24	0.31 ± 0.22	0.43 ± 0.27	0.38 ± 0.32

Table 5: 2_CNN_parallel & 2_HistCNN_parallel

	2_CNN_par Avg. F1-score	**2_HistCNN_par Avg. F1-score**	**2_CNN_par Avg. Precision**	**2_HistCNN_par Avg. Precision**	**2_CNN_par Avg. Recall**	**2_HistCNN_par Avg. Recall**
Sample A	0.44 ± 0.26	0.31 ± 0.26	0.45 ± 0.24	0.31 ± 0.31	0.48 ± 0.31	0.36 ± 0.31
Sample B	0.44 ± 0.24	0.31 ± 0.27	0.48 ± 0.23	0.31 ± 0.30	0.46 ± 0.30	0.36 ± 0.30
Sample C	0.45 ± 0.24	0.38 ± 0.25	0.46 ± 0.24	0.39 ± 0.30	0.49 ± 0.32	0.43 ± 0.30
Sample D	0.42 ± 0.28	0.31 ± 0.26	0.43 ± 0.24	0.35 ± 0.31	0.48 ± 0.33	0.38 ± 0.30

work. There are three pairs of networks each being comprised of 1 layer, 2 layers, and 2 parallel layers. Within each pair, we constrained the number of parameters to be as close as possible to evaluate the histogram CNN and standard CNN under similar architectures.

2.3. Training Procedure

For our training procedure, our data is first scaled to between 0 and 1. Next, we wanted to look at the effects of preprocessing our data through standardization

$$Z = \frac{x - \mu}{\sigma}. \tag{3}$$

The Equation 3 represents the standardization operation, where x is the original feature value, μ is the mean of that feature value, σ is its standard deviation, and Z is the resulting standardized data. For our dataset, x is the RGB value for each pixel and the corresponding mean (μ) and standard deviation (σ) for each channel will be used to normalize the data through Equation 3 . Data preprocessing is considered to be one of the most important components of training the model, and we want to compare the results from standardizing the data and relating this to the performance of each network.

For the training process, we use the same data augmentation and optimization technique for the six different neural networks shown in figure 1. Similar to [15], all the samples were first resized to 128 $\times$ 128 then a crop of random size (0.8 to 1.0) of the original size and a random aspect ratio (3/4 to 4/3) of the original aspect ratio is extracted. This crop is finally resized to 112$\times$112 and horizontal flips ($p = 0.5$) were used. The experiment starts with learning rate of 0.001 and batch size of 128. Throughout 100 total epochs, the learning rate decays by factor of 0.9 if model does not make significant changes within 10 epochs(threshold = 1e-4). During the learning rate decay, Adam, an adaptive learning rate optimization algorithm, is used. Adam allows for efficient stochastic optimization that only requires first-order gradients, allowing for little memory requirement[4]. We trained on one sample and then tested on the other three.

3. Result

After all the scripts were executed in HiPerGator, University of Florida's cluster research supercomputer

[10], data were collected and analyzed with our own python scripts. Shown in Figure 6, average overall accuracy was computed for each network both for the networks that used standardized data and non-standardized data. The networks that used pre-processed data averaged 6.50% better than networks that did not involve data-standardization in performance. As it was shown that networks trained with standardized data had better performance, the rest of the analysis was only focused on results from these experiments. Additionally, the average F1-score, precision, and recall value was calculated to evaluate each network's performance more deeply. Tables 3 through 5 each shows these collected values and compares between each standard CNN and histogram layer CNN pair for each sample runs. The figures 4 through 6 shows the average confusion matrices for each model. The confusion matrices for each model in each sample run is attached in the appendix marked as figure 10 through 12 Overall, all the networks showed strong performance in classifying brown_bread, corduroy, and cork classes, which is shown with darker color in the corresponding blocks from the figure. On the other hand, most of the networks struggled on classifying aluminum_foil, cotton, and white_bread.

Along with confusion matrices, the histograms learned by the CNNs with histogram layer(s) can be found in the appendix (figures 8 through 9). The number of bins are set to four, each curve in each figure represent a single bin, and the graph itself represents the bin center and bin width learned by each network for each sample run.

4. Discussion

4.1. Standardization

As mentioned above, the models were trained with standardized data and with non-standardized data. Table 6 shows the average overall accuracy for six networks with columns divided into whether data was normalized or not. Normalization for data-processing is considered to be very important when training the model, and we were able to evaluate how the performance differs by the methods used for data-preprocessing. As shown in Table 6, every network improved performance significantly with standardization, averaging about 6.50% increase in average overall accuracy. From this result, we learned how normalizing the data before training can improve performance compared to using non-normalized dataset. In future works, standardization will continue to be used as a pre-processing technique for our data.

4.2. Histogram Layer

Based on Table 6, 2_CNN_parallel had the highest average overall accuracy followed by 1_CNN. 2_HistCNN showed the highest performance from the histogram CNNs. Most of the histogram networks showed lower accuracy than the corresponding CNN except for 2HistCNN using unstandardized data. However, since this is a multi-class problem, average overall accuracy alone cannot determine networks' performance. Therefore, we looked at average f1-score, precision, and recall shown in Table 3 through 5 for further evaluation.

From table 3 through 5, standard CNNs showed higher value both on average precision and recall than histogram CNNs, indicating that standard CNNs had lower false positives and more true positives. This ultimately led standard CNNs to have a higher average f1 score value. Table 4 compares 2_CNN and 2_HistCNN. 2_CNN had mostly lower average f1-score, precision, and recall than 1_CNN's, but 2_HistCNN's f1-score, precision, and recall values were slightly higher than 1_HistCNN. This is most likely because of an additional histogram layer. For 2_CNN_parallel and 2_HistCNN_parallel shown in table 5, f1-score, precision, and recall values were mostly similar with data from 1_CNN and 1_HistCNN.

From these comparisons, we learned that the CNNs implemented with histogram layer did not necessarily perform better on texture classification. This may be due to insufficient tuning of parameters such as window size, kernel size, intialization method for histogram layer, and number of bins.

4.3. Confusion Matrices

Confusion matrices shown from figure 4 to 6 reflects how different network performed in specific classes. The darkness of blue shade is proportional to model's performance in that class, and dark blue blocks should align diagonally (top left to bottom right) in an ideal situation. In general, the networks performed very well on brown_bread, corduroy, and cork class but performed poorly on cotton, linen, and aluminum foil classes. By observing the distribution of the blue shades for other classes, there were focused false positive results for specific classes. Most of the networks, especially the non-histogram CNNs, predicted that the aluminum foil class was cork. On the other hand, classes such as cotton, cracker, lettuce leaf, and linen had more spread out false positive predictions unlike aluminum foil and wood classes.

Although most of the histogram CNNs did not necessarily perform better than the standard CNNs, they did show a better accuracy for some classes. For figure 4, 1HistCNN performed better on cotton class. For figure 5, 2HistCNN performed better on five classes (brown_bread, corduroy, cork, wood, and linen). Lastly, for figure 6, 2HistCNN_parallel performed better on three classes

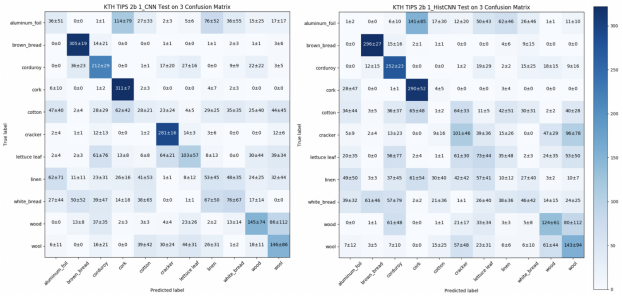


Figure 4: Confusion Matrices for 1_CNN (left) and 1_HistCNN (right)

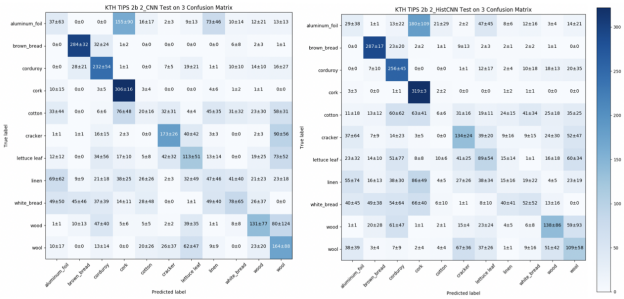


Figure 5: Average Confusion Matrices for 2_CNN (left) and 2_HistCNN (right)

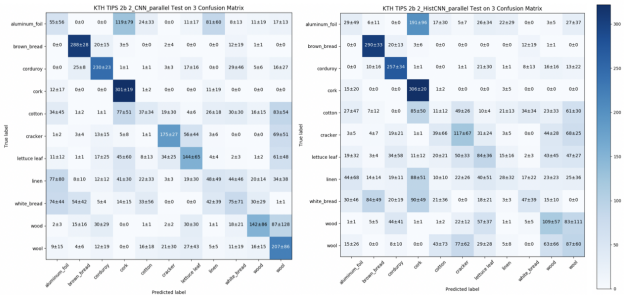


Figure 6: Average Confusion Matrices for 2_CNN_parallel (left) and 2_HistCNN_parallel (right)

(Brown_bread, corduroy, and cork). As these results suggest, histogram CNN showed stronger performance and confidence level on some classes.

The choice of which sample to use as testing set affected networks' performance in different classes as well. Although the blue shade distribution for cotton, cracker, lettuce leaf, and linen classes mostly looked random for A, B, and D test runs, networks trained with Sample C had more concentrated false positive predictions. Additionally, networks trained with Sample D had a similar effect on wood class. Unlike the other networks tested with A, B, and C sample, networks trained with Sample D showed strong confidence in predicting that the wood class texture images

Table 6: Average Overall Accuracy

Network	Average Overall Accuracy	
	Non-Standardized	Standardized
1_CNN	38.87 ± 4.05	47.48 ± 0.97
1_HistCNN	34.90 ± 4.09	37.43 ± 3.33
2_CNN	29.51 ± 5.14	44.35 ± 1.07
2_HistCNN	36.75 ± 2.58	40.14 ± 1.49
2_CNN_parallel	40.31 ± 3.79	47.66 ± 1.22
2_HistCNN_parallel	35.91 ± 4.15	38.20 ± 2.85

are wool. Sample D's data may have capured better information, such as illumination, pose, and scale, for that class.

From these analysis, we found that selection of which samples to use for training set affected the distribution of false positive results.

5. Conclusion

In this paper, we have introduced a stackable histogram layer that involves localized binning operation on CNN. Based on analysis of overall accuracy, f1-score, precision, recall, and confusion matrix, histogram layer CNNs did not necessarily have better performance. However, some histogram layers performed better than CNNs on specific classes. This may be due to incomplete parameters tuning done prior to the experiments. There are still several different parameters that need to be considered such as the initialization method, number of bins, window size, and kernel size. For example, we used four bins, which may have not been enough for this experiment. Therefore, in future works, more parameters can be considered to determine if better performance will occur in these ANN architectures

6. Acknowledgement

I express my deepest and sincere gratitude to my mentor and UF graduate student Joshua Peeples who personally supervised, supported and encouraged me through the whole work. He supported me from learning basic concepts and math for machine learning to calculating torch size and tuning parameters. I would also like to thank him for proofreading this paper, allowing me to submit the best final version for presentation. He allowed me to get the most out of my summer research experience and greatly influenced my interest in computer science/machine learning field.

I would also like to thank University of Florida's Student Science Training Program (UF SSTP) and my lab host Dr. Alina Zare for making this research experience possible. Over the course of seven-weeks, I participated in this research in Dr. Alina Zare's Machine Learning and Sensing Lab. Working with other lab members and participating in

weekly lab meetings helped me to understand the real university environment and this field more intuitively.

Lastly, I acknowledge University of Florida Research Computing for providing computational resources and support that have contributed to the research results reported in this publication. URL: http://researchcomputing.ufl.edu

References

[1] Saikat Basu, Supratik Mukhopadhyay, Manohar Karki, Robert DiBiano, Sangram Ganguly, Ramakrishna Nemani, and Shreekant Gayaka. Deep neural networks for texture classification—a theoretical analysis. *Neural Networks*, 97:173 – 182, 2018.

[2] P. Cavalin and L. S. Oliveira. A review of texture classification methods and databases. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pages 1–8, Oct 2017.

[3] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973.

[4] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[5] P. Mallikarjuna, Alireza Tavakoli Targhi, Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh. The kth-tips 2 database. 2006.

[6] Andrzej Materka and Michal Strzelecki. Texture analysis methods – a review. Technical report, INSTITUTE OF ELECTRONICS, TECHNICAL UNIVERSITY OF LODZ, 1998.

[7] Jakub Nalepa, Janusz Szymanek, Michael P. Hayball, Stephen J. Brown, Balaji Ganeshan, and Kenneth Miles. Texture analysis for identifying heterogeneity in medical images. In Leszek J. Chmielewski, Ryszard Kozera, Bok-Suk Shin, and Konrad Wojciechowski, editors, *Computer Vision and Graphics*, pages 446–453, Cham, 2014. Springer International Publishing.

[8] Dat Nguyen, Tuyen Pham, Na Rae Baek, and Kang Ryoung Park. Combining deep and handcrafted image features for presentation attack detection in face recognition systems using visible-light camera sensors. *Sensors*, 18:699, 02 2018.

[9] Rahul Nijhawan, Josodhir Das, and Balasubramanian Raman. A hybrid of deep learning and hand-crafted features based approach for snow cover mapping. *International Journal of Remote Sensing*, 40(2):759–773, 2019.

[10] University of Florida. Research computing.

[11] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[12] L. S. Pinto, A. Ray, M. U. Reddy, P. Perumal, and P. Aishwarya. Crop disease classification using texture analysis. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 825–828, May 2016.

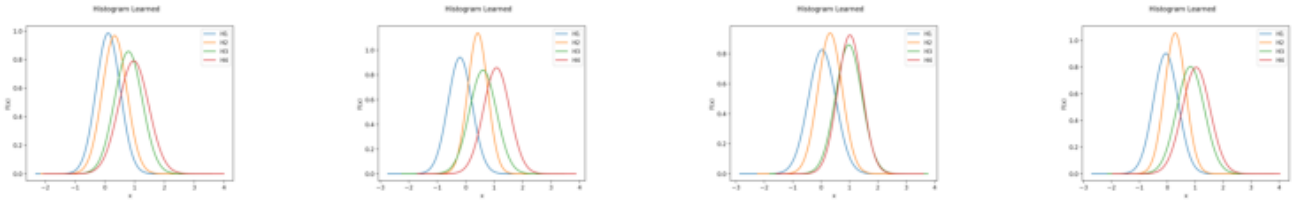
[13] Vahid Sedighi and Jessica Fridrich. Histogram layer, moving convolutional neural networks towards feature-based steganalysis. *Electronic Imaging*, 2017:50–55, 01 2017.

[14] Zhe Wang, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learnable histogram: Statistical context features for deep neural networks. *CoRR*, abs/1804.09398, 2018.

[15] Jia Xue, Hang Zhang, and Kristin J. Dana. Deep texture manifold for ground terrain recognition. *CoRR*, abs/1803.10896, 2018.

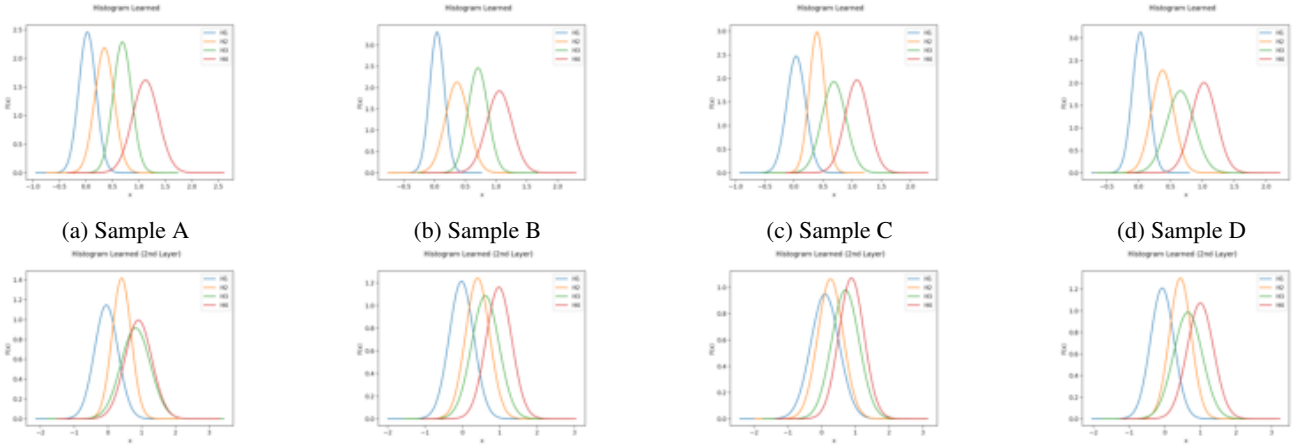
[16] Hang Zhang, Jia Xue, and Kristin J. Dana. Deep TEN: texture encoding network. *CoRR*, abs/1612.02844, 2016.

7. Appendix



(a) Sample A (b) Sample B (c) Sample C (d) Sample D

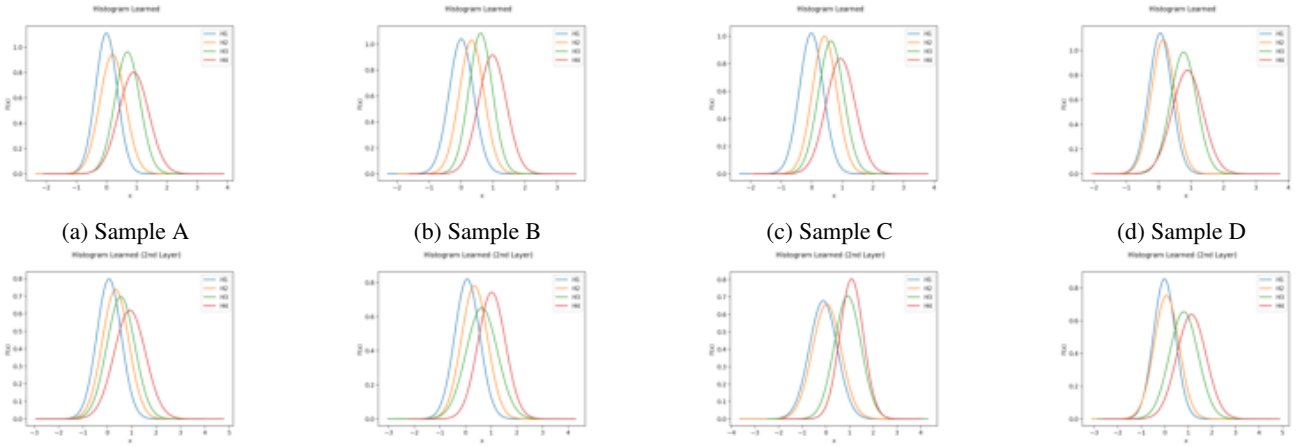
Figure 7: Histograms learned for 1HistCNN



(a) Sample A (b) Sample B (c) Sample C (d) Sample D

(e) Sample A (f) Sample B (g) Sample C (h) Sample D

Figure 8: Histograms learned for 2_HistCNN top row: 1st histogram layer (8a - 8d); bottom row: 2nd histogram layer (8e - 8h)



(a) Sample A (b) Sample B (c) Sample C (d) Sample D

(e) Sample A (f) Sample B (g) Sample C (h) Sample D

Figure 9: Histograms learned for 2_HistCNN_parallel top row: 1st histogram layer (9a - 9d); bottom row: 2nd histogram layer (9e - 9h)

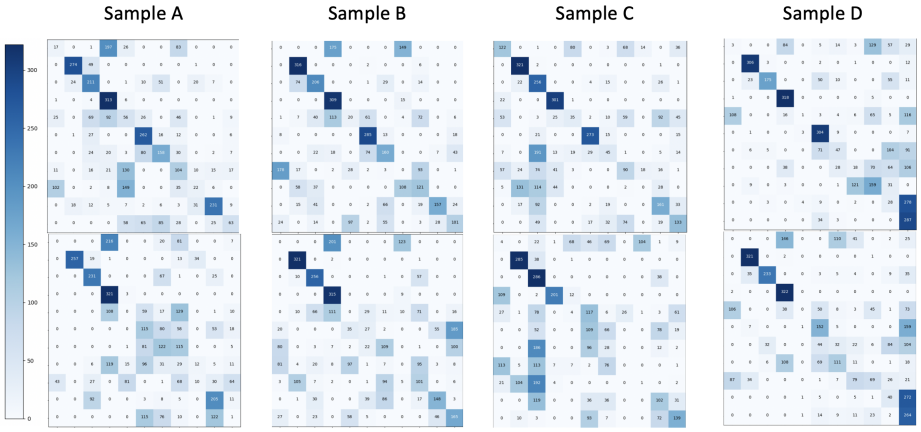


Figure 10: Confusion Matrices for 1_CNN (top row) and 1_HistCNN (bottom row)

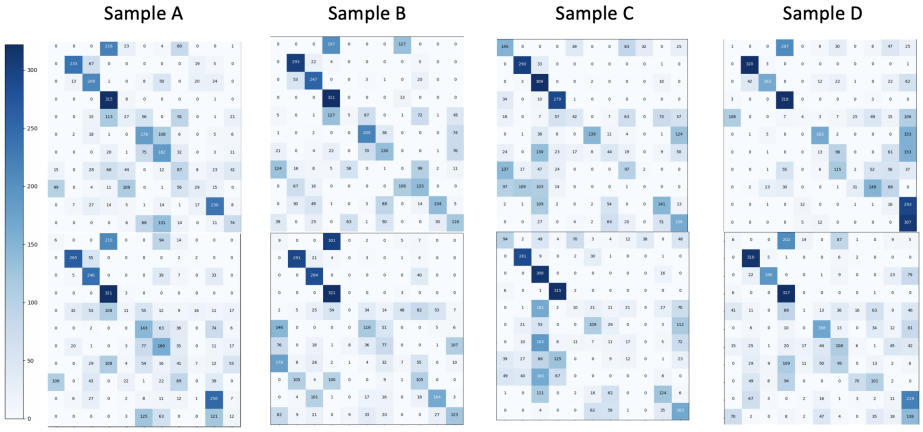


Figure 11: Confusion Matrices for 2_CNN (top row) and 2_HistCNN (bottom row)

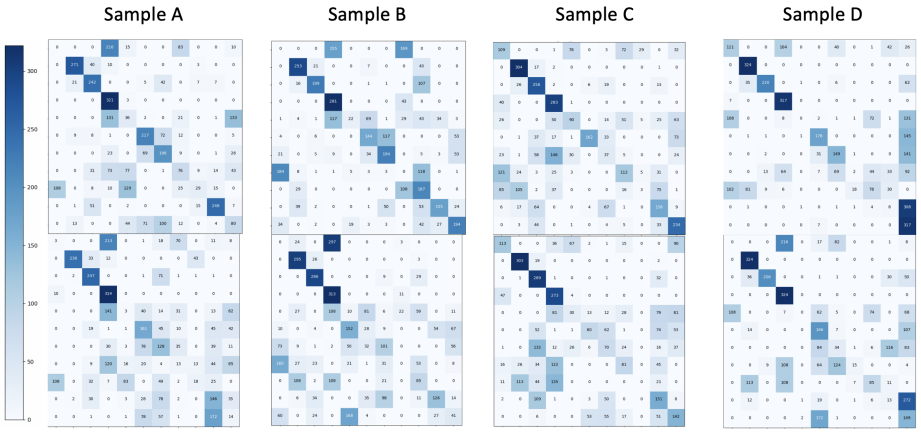


Figure 12: Confusion Matrices for 2_CNN_parallel (top row) and 2_HistCNN_parallel (bottom row)