Histogram Layers for Texture Analysis

Joshua Peeples, Weihuang Xu, and Alina Zare

Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611, USA
{jpeeples, weihuang.xu, azare}@ufl.edu

Abstract. We present a histogram layer for artificial neural networks (ANNs). An essential aspect of texture analysis is the extraction of features that describe the distribution of values in local spatial regions. The proposed histogram layer directly computes the spatial distribution of features for texture analysis and parameters for the layer are estimated during backpropagation. We compare our method with state-of-the-art texture encoding methods such as the Deep Encoding Network Pooling (DEP) [43], Deep Texture Encoding Network (DeepTEN) [46], Fisher-vector convolutional neural network (FV-CNN) [13], and Multi-level Texture Encoding and Representation (MuLTER) [20] on three material/texture datasets: (1) the Describable Texture Dataset (DTD) [12]; (2) an extension of the ground terrain in outdoor scenes (GTOS-mobile) [43]; (3) and a subset of the Materials in Context (MINC-2500) dataset [6]. Results indicate that the inclusion of the proposed histogram layer improves performance. The source code for the histogram layer is publicly available [1] .

Keywords: Texture analysis, deep learning, histograms

1 Introduction

Texture analysis is a crucial component in many applications including autonomous vehicles [14], automated medical diagnosis [9], and explosive hazard detection [3]. The concept of texture is easily discernible for humans, but there is no agreed definition within the computer vision community [39, 23]. Generally, variations in the definition of texture arise because of differences in the application being studied (*i.e.*, texture characteristics that are more informative vary across application areas) [39, 23]. Yet, most agree that one common component of texture analysis relies on characterizing the spatial distribution of intensity and/or feature values.

A number of handcrafted features have been developed with successful application to texture-dependent computer vision problems. However, the process to design these features can be difficult. Feature engineering is an expensive process in terms of labor, computation, and time and often required significant domain knowledge and expertise. Additionally, these features often rely on empirically determining the best parameters for each descriptor resulting in an increase of computation and time. For example, histogram-based features such as histogram of oriented gradients (HOG) [15]

[1] Code submitted in supplementary material

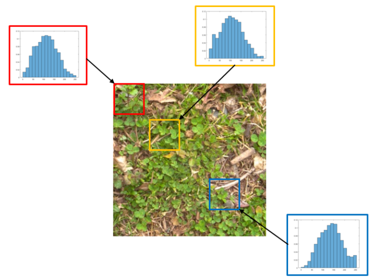


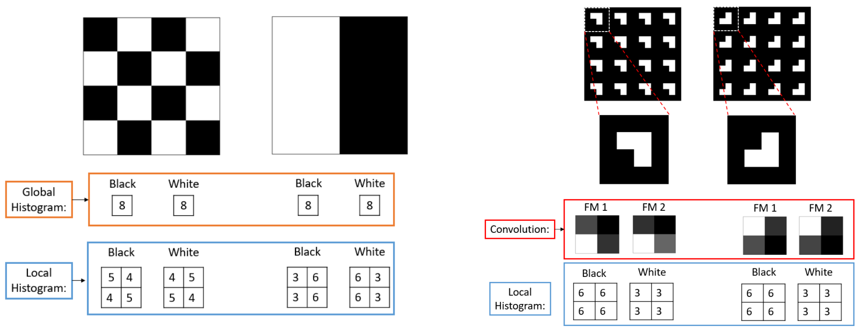
Fig. 1: This is an example image of grass from GTOS-mobile [43]. The image contains other textures and not only grass. Local histograms can distinguish portions of the image containing pure grass (top two histograms) or a mixture of other textures (bottom histogram). Integrating a histogram layer in deep neural networks will assist in estimating the data distribution to improve texture analysis.

and local binary patterns (LBP) [33] have been extensively studied and used in texture-based applications [28, 16, 32, 44]. In both HOG and LBP feature sets, spatial distributions of feature values are used to characterize and distinguish textures. Furthermore, the distributions are summarized using histograms.

In recent work, handcrafted feature extraction has often been substituted with automated feature learning using deep learning to address some of the issues associated with designing features. Deep learning has often outperformed approaches that couple hand-designed feature extraction with classification, segmentation and object recognition [18, 21, 22, 27]. Despite the success of deep learning, some works have shown empirically and theoretically that traditional features perform better or comparable to that of deep learning methods in texture analysis [4, 5, 10, 25]. Additionally, these deep learning models cannot model the distribution of values in regions which is essential for texture analysis [39]. Deep architectures require more layers resulting in more parameters to characterize the spatial distribution of features in a convolutional neural network (CNN) as opposed to using a histogram directly.

The proposed solution, a *histogram layer* for artificial nerual networks (ANNs), automates the feature learning process while simultaneously modeling the distribution of features. The histogram layer is a tool to integrate and utilize the strengths of both handcrafted features and deep learning to maximize texture analysis performance. Histograms are an effective and efficient approach to aggregate information. The selection of the bin centers and widths are crucial for the feature representation. Instead of manually determining these parameters, these parameters are estimated through backpropagation. Radial basis functions (RBFs) are used as the histogram binning operation to allow for the gradient information to flow through the network. The contributions of this work are:

- First localized histogram layer for texture analysis which maintain spatial context
- Bin centers and widths of the histogram estimated through backpropagation
- Robust to ambiguity and outliers through the use of “soft” bin assignments



(a) Local vs global histogram

(b) Local histogram vs convolution

Fig. 2: Toy 4×4 images showing the disadvantages of a global histogram (2a) and convolution operation (2b). On the left in 2a, the two images are distinct textures. If a global histogram is computed, the distribution of white and black pixels is equivalent resulting in no discrimination between the two texture types. On the right in 2b, convolution operations are sensitive to image transformations such as rotations. The two textures shown are the same, but applying filters from a convolutional neural network such as ResNet18 in local areas results in different feature maps (FM). However, a local histogram provides some rotational invariance and learns the same local distribution of pixels for each image.

2 Related Work

Deep Learning for Texture Analysis Deep learning has been used for texture applications [23, 10]. Attempts to combine neural and traditional features into deep learning architectures have shown success [34, 42, 40, 35], but the traditional features can not be updated through this process. Also, some have tried to emulate handcrafted features via the network design [8, 11, 30] but have run into issues including high computational costs and a decrease in texture analysis performance [23]. Another approach for texture analysis is to aggregate the features extracted by pre-trained models through "encoding layers" [13, 46, 37, 43]. As noted by Liu et al., these "encoding layers" have primarily focused on transfer learning approaches for convolutional neural networks (CNNs), but CNN features are sensitive to image transformations [24, 26] such as rotations as shown in Figure 2. The proposed histogram layer will be more robust than previous methods "encoding layers" due to "soft" binning assignments that are less sensitive to ambiguity and outliers in the feature maps. The proposed histogram layer can also be jointly trained with the convolutional layers to influence the features learned by the network.

Pooling Operations Common components of deep learning frameworks are pooling layers such as max pooling which captures the highest feature values and average pooling which computes the mean of each descriptor. Pooling layers provide several advantages such as generalization, reduced computational costs, and improved performance [17, 2]. However, these pooling layers make assumptions about the data that are not optimal for every dataset [2]. For example, some data types (such as synthetic aperature sonar imagery) are plagued with difficult-to-remove speckle noise [1]. The use of min or max pooling will tend to propagate noise values as opposed to more informative values.

Also, several pooling operations (*e.g.*, max pooling) only backpropagate the error through certain locations resulting in a saturation issue that slows learning [45]. The proposed histogram layer will retain the advantages of standard pooling operations but will learn the bin centers and widths necessary to aggregate the features of the data throughout the data distribution. The proposed histogram layer will also be robust to outliers in the data. If a value is far from each bin center, the contribution of the outlier will be negligible. Also, the proposed histogram layer also provides normalization of the features because the contribution of each descriptor for each bin is between the ranges of 0 and 1.

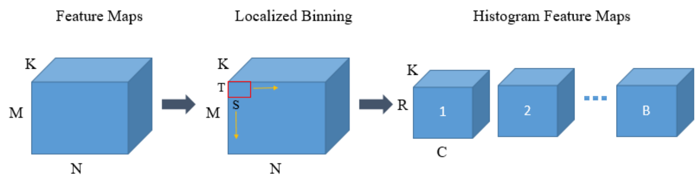
Previous Histogram Layers The standard histogram operation counts the number of values that fall within certain ranges. The center of these ranges are defined as "bin centers" and the interval or size of each range (or "bin") is defined by the "bin width." The standard histogram operation can be formulated as an indicator function. Consequently, this standard histogram operation cannot be easily used in ANNs because the functional form does not have the operation in terms of the histogram parameters (*i.e.*, the bin centers and widths) and is not differentiable thus cannot be directly updated via backpropagation [41]. In order to overcome the shortcomings of the standard histogram operation, two histogram layers were proposed for applications other than texture analysis. The first histogram layer was developed for semantic segmentation and object detection [41] by Wang, et al. The histogram operation was completed using a linear basis function to backpropagate the error to learn bin centers and widths. Wang et al.'s histogram layer has a convenient property in that it is implemented using pre-existing layers. The second histogram layer was developed for steganalysis [36] and the histograms were modeled using RBFs. Sedighi and Fridich did not update the bin centers and widths, but these values were fixed to simulate the submodels of the projection spatial rich model (PSRM) [19].

The histogram layer proposed in this work inherits properties from each of these models, but also incorporates novel aspects for texture analysis. The histogram layer will use RBFs to represent the histogram structure and this will provide smooth functions to update the bin centers and widths of the model. There are three key differences between our histogram layer and its predecessors. 1) Each of the previous approaches constructed global histograms. Spatial relationships are important in applications involving texture [31, 38] as shown in Figure 2 and a localized approach will retain this information. 2) The number of bins is varied as opposed to the previous methods that used only a single bin number. 3) The histogram layer can be placed anywhere in a network.

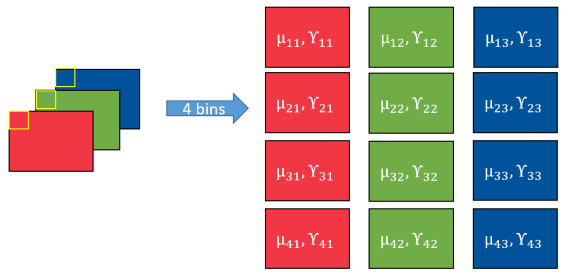
3 Proposed Histogram Layer

3.1 Soft Binning Operation

We model our histogram with RBFs [36]. RBFs provide smoother approximations for histograms and RBFs have a maximum value of 1 when the feature value is equal to the



(a) Visualization of localized histogram operation.



(b) Example of binning process with 3-channel input (RGB).

Fig. 3: Visualization of localized histogram operation is shown in Figure 3a. For the histogram layer, the input is K feature maps with spatial dimensions of $M \times N$. The normalized frequency count, Y_{rcbk}, can be computed with a sliding window of size $S \times T$ resulting in B histogram feature maps of size $R \times C \times K$ where B corresponds to the number of bins, R, C, and K are the resulting output dimensions after binning the feature maps. In Figure 3b, an example 3-channel input is passed into a histogram layer with 4 bins. The output of the histogram layer is 12 histogram feature maps, where each feature map is a binned output with a distinct bin center (μ_{bk}) and width (γ_{bk}) for each kth input channel.

bin center and the minimum value approaches 0 as the feature value moves further from the bin center. Also, RBFs are robust to small changes in bin centers and widths than the standard histogram operation because there is some allowance of error due to the soft binning assignments and the smoothness of the RBF. The means of the RBFs (μ_{bk}) serve as the location of each bin (*i.e.*, bin centers) while the bandwidth (γ_{bk}) controls the spread of each bin (*i.e.*, bin widths). The normalized frequency count, Y_{rcbk}, is computed with a sliding window of size $S \times T$ and the binning operation for a histogram value in the k^{th} channel of the input x is defined as:

$$Y_{rcbk} = \frac{1}{ST} \sum_{s=1}^{S} \sum_{t=1}^{T} e^{-\gamma_{bk}^2 (x_{r+s,c+t,k} - \mu_{bk})^2} \tag{1}$$

where r and c are spatial dimensions of the histogram feature maps. The process of aggregating the feature maps is shown in Figure 3.

Backpropagation The histogram layer supports end-to-end learning through backpropagation to update the bin centers and widths. Each k^{th} channel of the input x is binned by the histogram in local spatial regions and stored in the r^{th} row and c^{th} column of the output of the histogram layer, Y_{rcbk}. The gradients for the parameters of the histogram layer with a window size of $S \times T$ are computed by Equations 2 and 3:

$$\frac{\partial Y_{rcbk}}{\partial \mu_{bk}} = \frac{2}{ST} \sum_{s=1}^{S} \sum_{t=1}^{T} e^{-\gamma_{bk}^2 (x_{r+s,c+t,k} - \mu_{bk})^2} \gamma_{bk}^2 \left(x_{r+s,c+t,k} - \mu_{bk} \right) \tag{2}$$

$$\frac{\partial Y_{rcbk}}{\partial \gamma_{bk}} = \frac{-2}{ST} \sum_{s=1}^{S} \sum_{t=1}^{T} e^{-\gamma_{bk}^2 (x_{r+s,c+t,k} - \mu_{bk})^2} \gamma_{bk} \left(x_{r+s,c+t,k} - \mu_{bk} \right)^2 \tag{3}$$

where $\frac{\partial Y_{rcbk}}{\partial \mu_{bk}}$ and $\frac{\partial Y_{rcbk}}{\partial \gamma_{bk}}$ are partial derivatives of Y_{rcbk} with respect to the bin centers and widths of the histogram layer. In [36], the tails of the RBFs were set to 1 resulting in the gradient becoming zero if the feature map value is outside of every bin centers range. In our histogram layer, the gradients are a function of the distance between the feature map value and the bin centers. The gradient contribution from a feature map value will be small if it is far away from every bin center (*i.e.*, outliers).

3.2 Implementation

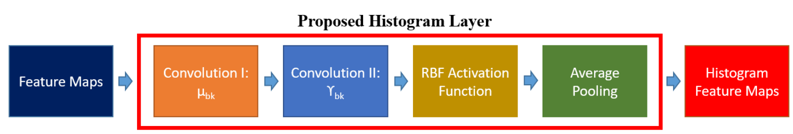


Fig. 4: Histogram layer implementation using pre-existing layers.

The histogram layer is implemented using commonly used pre-existing layers as shown in Figure 4. As done in [46, 43], a 1x1xK convolution is used to reduce the number of input feature maps where K is the new dimensionality of the feature maps. After the dimensionality reduction, the binning process starts by first assigning each feature value to every bin center (subtracting μ_{bk}). The centering of the features to each bin is calculated by applying a 1x1xB convolution to each feature map. The weights in the convolution kernels are fixed to 1 and each bias serves as the learnable bin centers.

After the features are assigned to the bins, the centered features are then multiplied by the bandwidth (γ_{bk}) to incorporate the spread of the features for each bin. The incorporation of the spread of each bin is also computed by applying a 1x1xB convolution to each feature map with the weights serving as the learnable bin widths and fixing the biases to be 0. The contribution to each bin is calculated through RBF activation functions in Equation 1 by squaring, negating, and applying the exponential to the centered and scaled features. The contribution of each feature to every bin is between 0 and 1. The contributions of features in local spatial regions are then counted through average pooling to compute the normalized frequency count of features belonging to each bin.

4 Experimental Procedure

4.1 Experimental Setup

We performed material/texture classification with the proposed histogram layer in comparison to other texture encoding approaches. The experiments consisted of two components: 1) configuration of histogram layer and 2) comparison to other texture encoding approaches. For the first component, we investigated the effects of a) normalization , b) sum-to-one constraint across bins, and c) location of histogram layer. Previous works incorporated batch normalization of their convolutional features before fusing their texture features (*i.e.*, concatenation, weighting) [20, 43, 46]. We wanted to investigate whether the impact of batch normalization for our new layer would improve results similar to existing "encoding layers." In normalized histograms, the heights of the bins (*i.e.*, counts within each bin) are constrained to sum up to one across each bin. We also wanted to compare if including or relaxing this constraint would effect overall performance and/or increase the robustness of the histogram layer. The final part of the configuration of the histogram layer was the effect of location in the networks. The histogram layer was added at the end of the network for the normalization and sum-to-one constraint experiments, but it may be more advantageous to learn texture information earlier in the network. Previous histogram-based features used lower level features (*e.g.*, edges [16] for edge histogram descriptors) and learning histograms of convolutional features earlier in the network may exploit more texture information.

Training Details A similar training procedure from [20, 43] was used in this work. For each dataset, the image is resized to 256×256 and a random crop of 80 to 100 % of the image was extracted with a random aspect ratio of 3/4 to 4/3. The crop was then resized to 224×224 and the images were normalized by subtracting the per channel mean and dividing by the per channel standard deviation. A random horizontal flip ($p = .5$) was also added for data augmentation. The training settings for each network were the following: batch size of 128 (64 for the DTD and MINC-2500 models in the scale experiments), cross-entropy loss function, SGD with momentum ($\alpha = .9$), learning rates decay every 10 epochs by a factor of .1 and training is stopped after 30 epochs. The initial learning rates for the newly added and pre-trained layers were .01 and .001 respectively.

Architectures and Evaluation Metrics Two pre-trained ResNet models, ResNet18 and ResNet50, were used as the baseline for the convolutional features. The models that incorporated the histogram layer are referred to as HistRes_B, where B is the number of bins. The HistRes_B architecture is shown in Figure 5. For the normalization and sum-to-one constraint experiments, the number of bins was also varied to investigate the effects of adding additional histogram feature maps to the network. After the normalization and sum-to-one constraint experiments were completed, the placement of the histogram layer in the each model was investigated. The kernel size and stride for the histogram was selected to produce 2×2 local feature maps for each location so that the number of histogram features were equal to the number of convolutional features from the global average pooling (GAP) layer (512 and 2048 for ResNet18 and ResNet50 respectively).

For the channel-wise pooling, the value of K was selected so that the number of features from the GAP layer and the histogram layer was equal in order to leverage the contribution of texture, spatial and orderless convolutional features. For Resnet18, the input number of feature maps from each scale was reduced to 32, 16, and 8 for HistRes_4, HistRes_8, and HistRes_16 respectively. For Resnet50, the input number of feature maps from each scale was reduced to 128, 64, and 32 for HistRes_4, HistRes_8, and HistRes_16 respectively. The bin centers and widths were initialized to values sampled uniformly between $\left(\frac{-1}{\sqrt{BK}}, \frac{1}{\sqrt{BK}}\right)$. After the best configuration of the HistRes model was determined, the performance was compared using overall acccuracy between different "encoding layers" for each version of ResNet: 1) global average pooling (GAP), 2) Deep Texture Encoding Network (DeepTEN) [46], 3) Deep Encoding Pooling (DEP) [43], 4) Fisher-vector CNN (FV-CNN) [13] 5) Multi-level texture encoding and representation (MuLTER) [20] and 6) our HistRes_B. We also perform T-Tests to analyze the significance of the difference between results of comparison methods.

Datasets Three material/texture datasets were investigated: Describable Texture Dataset (DTD) [12], a subset of Materials in Context (MINC-2500) [6], and an extension of the ground terrain in outdoor scenes (GTOS-mobile) [43]. GTOS-mobile contains images for different resolutions (256×256, 384×384, and 512×512) but only single-scale experiments using 256×256 images were performed in this work. The total number of training and testing images for DTD, MINC-2500, and GTOS-mobile were 54,625, 5,640, and 37,381 respectively. The number of classes for DTD, MINC-2500, and GTOS-mobile were 47, 23, and 31 respectively. For DTD and MINC-2500, the published training and testing splits were used in each experiment (five and ten folds for DTD and MINC-2500 respectively). The ResNet50 architecture was used as the baseline model for these DTD and MINC-2500 while ResNet18 was used for GTOS-mobile [43]. GTOS-mobile only has a single training and test split, but five experimental runs were performed to investigate the stability of our model.

5 Results

5.1 Configuration of histogram layer

Normalization The first step of determining the best configuration of the histogram models was to investigate the effects of normalization. For the convolutional and histogram features, batch normalization and normalizing the count is used respectively. From Table 1, normalization of the features improved performance for each dataset, with the largest improvements occurring for the DTD dataset. The images in DTD are collected "in-the-wild" (*i.e.,* collected in a non-controlled environment) so normalization plays an important role in assisting the model to account for large intra- and interclass variations. For MINC-2500 and GTOS-mobile, the improvement did occur but to a lesser extent. Both datasets are also collected "in-the-wild," but each dataset is significantly larger than DTD and also have fewer number of classes, so normalization may not have lead to comparably large improvements like the models trained with DTD. Also, the number of bins did not affect performance for either the normalized or unnormalized models. This shows that the histogram layer performance does not rely heavily on the number of bins selected unlike conventional histogram-based feature approaches.

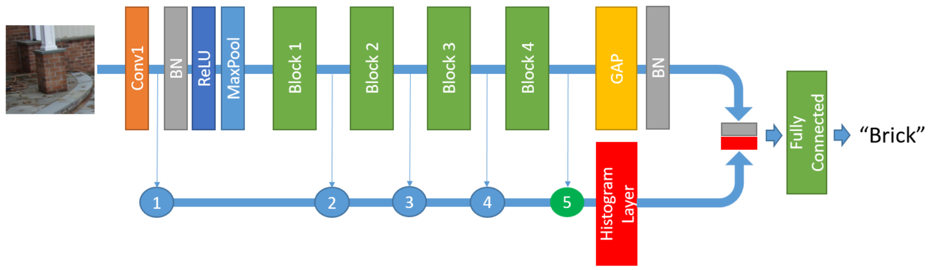


Fig. 5: Histogram Layer for ResNet with B bins (HistRes_B) based on ResNet18 and ResNet50 [18]. The convolutional features from the model are passed into the global average pooling (GAP) and histogram layer to capture texture, spatial and orderless convolutional features. The features are then concatenated together before being fed into a fully connected layer for classification. The location of the histogram layer was varied from 1 to 5. In this figure, the feature maps from the last convolution layer (location 5) are passed into the histogram layer.

Sum-to-one In addition to normalization, typical histograms bin votes are constrained to sum to one since histograms estimate probability density functions (PDFs) and PDFs integrate to one. In the normalization experiments, this constraint was relaxed. Ideally, relaxing this constraint will provide increased robustness to outliers. On the otherhand, this may also prevent learning if all the features initially start outside the range of the histogram (*i.e.*, vanishing gradient will occur if features are in the tails of the RBFs). As shown in Table 2, the enforcement of the constraint improved performance slightly for each dataset except for DTD. The model still retains robustness due to the soft binning assignments, but enforcing this constraint will assist in overcoming poor initialization by promoting each feature value to have some contribution to each bin.

Location of histogram layer For texture analysis, the location/scale at which features are extracted are important. Most texture "encoding layers" are placed at the end of the network. Intuitively, this makes sense because CNNs learn simple features such as edges and corners early in the network and then progress towards domain-specific information. The features extracted at the end of the network represent larger areas of the image (*i.e.*, larger scales). Since pre-trained models are used, "encoding layers" are placed at the end to tailor these features towards the application of interest. We wanted to verify this hypothesis by placing the best histogram model for each dataset at various locations in the network. For DTD, HistRes_4 with no sum-to-one constraint was used while both GTOS-mobile and MINC-2500 both used HistRes_16 with the sum-to-one constraint enforced. From Table 3, performance did not vary much with the location of the histogram layer. This shows that the histogram layer can use both simple and domain specific features to perform texture classification. As a result, this new layer can be placed anywhere in the network and achieve the same level of performance by learning the distribution of low to high level features. The number of features for each location of the histogram were constrained to be the same (512 and 2048 for ResNet18 and

Table 1: Test accuracy for each HistRes model 1a) with and 1b) without normalization for convolutional and histogram layer features. The result with the best average is bolded.

Dataset	HistRes_4	HistRes_8	HistRes_16
DTD	**71.98±1.23**	71.62±0.80	71.69±1.09
MINC-2500	82.14±0.31	81.97±0.47	**82.14±0.51**
GTOS-mobile	78.18±0.33	78.55±0.88	**78.56±0.71**

(a) Batch normalization and normalized count

Dataset	HistRes_4	HistRes_8	HistRes_16
DTD	62.58±3.27%	66.44±2.42%	68.07±1.86%
MINC-2500	80.03±1.23%	80.06±1.10%	81.38±1.08%
GTOS-mobile	72.31±2.93%	73.82±2.08%	74.48±2.37%

(b) No batch normalization and unnormalized count

ResNet50 respectively). An interesting future work would be to remove this constraint and use the same window size for each location in the network to further investigate the impact of the location of the histogram layer for performance.

5.2 Comparison to other texture encoding approaches

The histogram models for each dataset was compared to other state-of-the-art texture coding methods as well as the baseline ResNet models that use GAP. Overall, the histogram model performed better or comparably than GAP, DeepTEN and FV-CNN as shown in Table 4. For DTD, each model performed comparably. A reason for this is that a majority of the DTD dataset contains images with homogeneous textures and local information retained by the histogram layer may not provide significant additional information. However, when compared to DeepTEN, a method that focuses on orderless texture information, the addition of spatial information provided by the proposed histogram layer provides improved performance.

For MINC-2500, most images have the texture of interest in local regions of the image. The histogram model performed comparably to the other encoding approaches but again outperforming DeepTEN, further demonstrating the effectiveness of retaining spatial and texture information in the histogram layer. The DEP network also retains both spatial and texture information, but each HistRes_B achieved slightly better accuracy than the DEP model. The MuLTER [20] incorporates texture at multiple levels and the batch size for MuLTER was set to 32, resulting in the model seeing the data more times (in comparison to our batch size of 128). Despite these differences, our method achieves slightly better accuracy as well and there is no significant difference between the results obtained. As noted in [20, 43, 46], training the proposed models with 1) multiple images of different sizes and 2) varying scales in network will ideally lead to even more improved performance for our histogram layer in future works.

Table 2: Test accuracy for each HistRes model 2a) with and 2b) without sum-to-one constraint across bins. The result with the best average is bolded.

Dataset	HistRes_4	HistRes_8	HistRes_16
DTD	71.62±1.05%	71.53±1.21%	71.75±1.11%
MINC-2500	82.23±0.41%	**82.31±0.44%**	82.42±0.33%
GTOS-mobile	78.64±0.76%	78.77±0.81%	**79.75±0.84%**

(a) Enforced sum-to-one constraint across bins

Dataset	HistRes_4	HistRes_8	HistRes_16
DTD	**71.98±1.23%**	71.62±0.80%	71.69±1.09%
MINC-2500	82.14±0.31%	81.97±0.47%	82.14±0.51%
GTOS-mobile	78.18±0.33%	78.55±0.88%	78.56±0.71%

(b) Relaxed sum-to-one constraint across bins

Table 3: Test Accuracy for varying location of histogram layer for best histogram model for each dataset: HistRes_4 with no sum-to-one constraint for DTD and HistRes _16 with the sum-to-one constraint (all features for each model were normalized). The number of features for each location was constrained to be the same which lead to different window sizes and strides for each location. The kernel sizes for locations 1 through 5 were 64, 32, 16, 8 and 4 respectively. The strides for locations were 32, 16, 8, 4 and 2. The result with the best average is bolded.

Dataset	Location 1	Location 2	Location 3	Location 4	Location 5
DTD	71.28±1.03%	71.37±0.99%	71.16±1.02%	71.13±1.00%	**71.50±0.78%**
MINC-2500	**83.07±0.54%**	82.96±0.57%	82.96±0.27%	83.04±0.71%	**83.07±0.42%**
GTOS-mobile	78.63±1.00%	78.57±0.61%	78.97±0.82%	79.12±1.70%	**79.75±0.84%**

For GTOS-mobile, our proposed method achieves statistically significant performance in comparison to all other texture encoding methods as shown by the T-Tests in Table 4. GTOS-mobile contains the most natural, textured images such as grass and sand. Histogram-based features, as shown throughout the literature [23, 39], effectively represent texture features within natural imagery. The textures are also located in various spatial regions throughout the imagery. As a result, our localized histogram models are able to capture this information to improve performance.

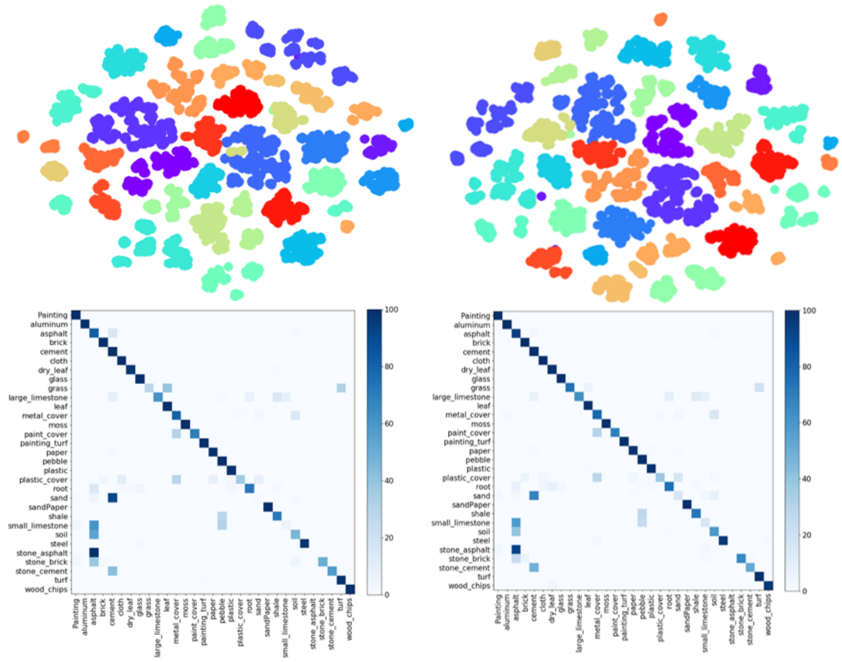
In Figure 6, the t-SNE visualization [29] and confusion matrices are shown for the GAP and best histogram model for GTOS-mobile. For the GAP model in Figure 6a, the t-SNE visual shows that most classes in the training data are clustered well but some samples are not near their same class and there is some overlap between neighboring classes. In Figure 6b, there is more separation between the classes and samples that belong to the same class appear to be closer to one another. To further validate our qualitative observation, the log of Fisher's Discriminant Ratio (FDR) [7] was computed to measure the ratio of separation and compactness for each class using the features from the 1) GAP and 2) HistRes_16. The quantitative values from the log of

Table 4: Test accuracy of each encoding method. The results for DeepTEN[46], DEP[43], and FV-CNN[13] are reported from [43] while MuLTER is reported from [20]. The results we obtained running the baseline ResNet model with global average pooling (GAP) are indicated by GAP* while GAP was reported from [43]. For our experiments, we average our results for each data split and show a 1-standard deviation to show the stability of our method. We compare the best histogram model: HistRes_4 for DTD and HistRes_16 for MINC-2500 and GTOS-mobile. The result with the best average is bolded. T-tests (one sample T-Test for each texture encoding approach except for GAP*) were performed between our HistRes_B model and the other methods. If the p-value was less than 0.05, than the difference in performance between our HistRes_B and comparison methods was considered statistically significant (indicated by ✓).

Dataset	GAP		GAP*		DeepTEN		DEP		FV-CNN		MuLTER		HistRes_B
DTD	-		73.07±0.79	✓	69.6	✓	**73.2**	✓	72.3	✗	-		71.98±1.23
MINC-2500	-		**83.01±0.38**	✗	80.4	✓	82.0	✓	63.1	✓	82.2	✗	82.42±0.33
GTOS-mobile	70.82	✓	76.09±0.91	✓	74.2	✓	76.07	✓	-		78.2	✓	**79.75±0.84**

Table 5: The log of the Fisher's Discriminant Ratio (FDR) [7] was computed on the features extracted from the 10,000 randomly sampled training images from the GTOS-mobile dataset (the same random images were used for each model). The FDR measures the ratio of the inter- (*i.e.*, between class) and intra- (*i.e.*, within class) class variance. Larger FDRs indicate that the samples within a class are close to each other and well separated from other samples in different classes. Overall and on a per-class basis, the separation of the features for our model is better. For 19 of the 31 classes, the log of the FDR is larger for the HistRes_16 model. Though the FDRs for the training data are better for HistRes_16 than GAP, our proposed model still generalized well to the test set in comparison to the baseline model (as indicated in Figure 6). The best average for the log of the FDR for each class is bolded.

Class Name	GAP	HistRes_16	Class Name	GAP	HistRes_16
Painting	37.42±1.44	**38.08±1.08**	pebble	41.78±1.71	**42.57±1.50**
aluminum	39.77±2.51	**39.90±1.85**	plastic	41.54±1.34	**41.91±1.00**
asphalt	7.37±0.13	**27.66±0.57**	plastic_cover	**39.3±1.27**	38.87±1.68
brick	36.09±1.03	**37.76±0.50**	root	37.58±1.07	**37.60±1.67**
cement	7.54±0.13	**25.21±2.44**	sand	**37.96±17.90**	35.73±16.64
cloth	**39.95±0.94**	36.28±1.28	sandPaper	**42.84±1.81**	41.79±2.22
dry_leaf	**40.19±3.09**	38.6±1.80	shale	41.11±1.20	**41.25±0.97**
glass	37.05±2.10	**38.34±1.26**	small_limestone	39.17±1.06	**39.19±1.07**
grass	39.02±1.42	**39.92±1.38**	soil	**39.09±0.83**	38.92±0.74
large_limestone	38.78±1.95	**39.14±1.15**	steel	39.71±1.61	**40.3±1.03**
leaf	**39.66±1.56**	38.77±1.77	stone_asphalt	42.91±0.90	**43.38±0.51**
metal_cover	37.64±1.79	**37.88±1.59**	stone_brick	**38.78±0.66**	38.76±1.11
moss	**38.99±1.78**	38.97±2.15	stone_cement	**42.32±1.26**	41.47±1.20
paint_cover	**41.43±0.41**	40.39±1.82	turf	17.24±20.19	**37.61±16.88**
paint_turf	38.57±0.79	**39.17±1.39**	wood_chips	**39.99±1.19**	39.64±1.66
paper	**41.20±1.14**	40.18±1.35	All Classes	39.95±6.33	**40.25±4.45**



(a) GAP (76.09±0.91%)

(b) HistRes_16 (ours, 79.75±0.84%)

Fig. 6: t-SNE [29] 2-D visualization of features extracted before the fully connected layer from 10,000 randomly sampled training images from the GTOS-mobile dataset (the same random images were used for each model) and the average test confusion matrices across the five runs of each model. The HistRes_16 retains the compact clusters but also adds some separation between classes. The confusion matrices are colored based on class percentages. Overall, the histogram model correctly classifies the various textures as well reducing the number of misclassifications.

FDR matches our qualitative observation from the t-SNE visualization. A reason for this is that the histogram features can be thought of as a similarity measure. If samples have similar textures, the distributions captured by the histogram layer should also be similar. Also, in the confusion matrices, the histogram model identifies the “sand” images while the GAP model does not. Overall, HistRes_16 improves performance for most classes and reduces the number of misclassified textures.

6 Conclusion

In this work, we presented a new layer for deep neural networks, a *histogram layer*. Previous deep learning approaches using CNNs are unable to effectively represent texture without adding more layers. Through the histogram layer, we directly capture texture information within images by characterizing the distribution of feature maps extracted from CNN models and can jointly fine-tune both the histogram layer and CNNs together. An in-depth analysis of the configuration of the histogram layer was conducted and provided more insight into the effects of normalization, sum-to-one constraints on the bins, and location of the histogram layer. The HistRes_B models were compared to other state-of-the-art texture encoding approaches and achieved statistically significant improvements in performance. The histogram layer can easily be integrated into other ANNs and used in other other texture analysis tasks besides classification such as segmentation, texture synthesis and shape from texture [23].

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1842473.

References

1. Abu, A., Diamant, R.: Robust image denoising for sonar imagery. In: 2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO). pp. 1–5. IEEE (2018)
2. Akhtar, N., Ragavendran, U.: Interpretation of intelligence in cnn-pooling processes: a methodological survey. Neural Computing and Applications pp. 1–20
3. Anderson, D.T., Stone, K.E., Keller, J.M., Spain, C.J.: Combination of anomaly algorithms and image features for explosive hazard detection in forward looking infrared imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **5**(1), 313–323 (2012)
4. Basu, S., Karki, M., Mukhopadhyay, S., Ganguly, S., Nemani, R., DiBiano, R., Gayaka, S.: A theoretical analysis of deep neural networks for texture classification. In: 2016 International Joint Conference on Neural Networks (IJCNN). pp. 992–999. IEEE (2016)
5. Basu, S., Mukhopadhyay, S., Karki, M., DiBiano, R., Ganguly, S., Nemani, R., Gayaka, S.: Deep neural networks for texture classification-a theoretical analysis. Neural Networks **97**, 173–182 (2018)
6. Bell, S., Upchurch, P., Snavely, N., Bala, K.: Material recognition in the wild with the materials in context database. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3479–3487 (2015)
7. Bishop, C.M.: Pattern recognition and machine learning. springer (2006)
8. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1872–1886 (2013)
9. Castellano, G., Bonilha, L., Li, L., Cendes, F.: Texture analysis of medical images. Clinical radiology **59**(12), 1061–1069 (2004)
10. Cavalin, P., Oliveira, L.S.: A review of texture classification methods and databases. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T). pp. 1–8. IEEE (2017)
11. Chan, T.H., Jia, K., Gao, S., Lu, J., Zeng, Z., Ma, Y.: Pcanet: A simple deep learning baseline for image classification? IEEE transactions on image processing **24**(12), 5017–5032 (2015)
12. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , Vedaldi, A.: Describing textures in the wild. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2014)

13. Cimpoi, M., Maji, S., Vedaldi, A.: Deep filter banks for texture recognition and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3828–3836 (2015)
14. Curio, C., Edelbrunner, J., Kalinke, T., Tzomakas, C., Von Seelen, W.: Walking pedestrian recognition. In: Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383). pp. 292–297. IEEE (1999)
15. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: international Conference on computer vision & Pattern Recognition (CVPR'05). vol. 1, pp. 886–893. IEEE Computer Society (2005)
16. Frigui, H., Gader, P.: Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-nearest neighbor classifier. IEEE Transactions on Fuzzy Systems **17**(1), 185–199 (2008)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
19. Holub, V., Fridrich, J.: Random projections of residuals for digital image steganalysis. IEEE Transactions on Information Forensics and Security **8**(12), 1996–2006 (2013)
20. Hu, Y., Long, Z., AlRegib, G.: Multi-level texture encoding and representation (multer) based on deep neural networks. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 4410–4414. IEEE (2019)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
22. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3367–3375 (2015)
23. Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., Pietikäinen, M.: From bow to cnn: Two decades of texture representation for texture classification. International Journal of Computer Vision **127**(1), 74–109 (2019)
24. Liu, L., Chen, J., Zhao, G., Fieguth, P., Chen, X., Pietikäinen, M.: Texture classification in extreme scale variations using ganet. IEEE Transactions on Image Processing (2019)
25. Liu, L., Fieguth, P., Guo, Y., Wang, X., Pietikäinen, M.: Local binary features for texture classification: Taxonomy and experimental study. Pattern Recognition **62**, 135–160 (2017)
26. Liu, L., Fieguth, P., Wang, X., Pietikäinen, M., Hu, D.: Evaluation of lbp and deep texture descriptors with a new robustness benchmark. In: European Conference on Computer Vision. pp. 69–86. Springer (2016)
27. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
28. Lowe, D.G., et al.: Object recognition from local scale-invariant features. In: iccv. vol. 99, pp. 1150–1157 (1999)
29. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
30. Malof, J.M., Bralich, J., Reichman, D., Collins, L.M.: Improving the histogram of oriented gradient feature for threat detection in ground penetrating radar by implementing it as a trainable convolutional neural network. In: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII. vol. 10628, p. 106280D. International Society for Optics and Photonics (2018)

31. Ojala, T., Pietikäinen, M.: Texture classification, machine vision, and media processing unit. University of Oulu, Finland (2004)
32. Ojala, T., Pietikainen, M., Harwood, D.: Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In: Proceedings of 12th International Conference on Pattern Recognition. vol. 1, pp. 582–585. IEEE (1994)
33. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis & Machine Intelligence (7), 971–987 (2002)
34. Paul, R., Hawkins, S.H., Hall, L.O., Goldgof, D.B., Gillies, R.J.: Combining deep neural network and traditional image features to improve survival prediction accuracy for lung cancer patients from diagnostic ct. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 002570–002575. IEEE (2016)
35. Rivera-Morales, C.A., Bastidas-Rodríguez, M.X., Prieto-Ortiz, F.A.: Densenet model combined with haralick's handcrafted features for texture classification. In: 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI). pp. 1–6. IEEE (2018)
36. Sedighi, V., Fridrich, J.: Histogram layer, moving convolutional neural networks towards feature-based steganalysis. Electronic Imaging **2017**(7), 50–55 (2017)
37. Song, Y., Zhang, F., Li, Q., Huang, H., O'Donnell, L.J., Cai, W.: Locally-transferred fisher vectors for texture classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4912–4920 (2017)
38. Srinivasan, G., Shobha, G.: Statistical texture analysis. In: Proceedings of world academy of science, engineering and technology. vol. 36, pp. 1264–1269 (2008)
39. Tuceryan, M., Jain, A.K.: Texture analysis. In: Handbook of pattern recognition and computer vision, pp. 235–276. World Scientific (1993)
40. Wang, H., Roa, A.C., Basavanhally, A.N., Gilmore, H.L., Shih, N., Feldman, M., Tomaszewski, J., Gonzalez, F., Madabhushi, A.: Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. Journal of Medical Imaging **1**(3), 034003 (2014)
41. Wang, Z., Li, H., Ouyang, W., Wang, X.: Learnable histogram: Statistical context features for deep neural networks. In: European Conference on Computer Vision. pp. 246–262. Springer (2016)
42. Wu, J., Lin, Z., Zha, H.: Multi-view common space learning for emotion recognition in the wild. In: Proceedings of the 18th ACM International Conference on Multimodal Interaction. pp. 464–471. ACM (2016)
43. Xue, J., Zhang, H., Dana, K.: Deep texture manifold for ground terrain recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2018)
44. Yang, Y., Newsam, S.: Bag-of-visual-words and spatial extensions for land-use classification. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems. pp. 270–279. ACM (2010)
45. Yu, D., Wang, H., Chen, P., Wei, Z.: Mixed pooling for convolutional neural networks. In: International Conference on Rough Sets and Knowledge Technology. pp. 364–375. Springer (2014)
46. Zhang, H., Xue, J., Dana, K.: Deep ten: Texture encoding network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 708–717 (2017)