Ensembling Framework for Texture Extraction (TE) Techniques for Classification

Vijay Pandey\textsuperscript{a,}\textsuperscript{*}, Mayank Gubba\textsuperscript{b}, Mohammed Faisal\textsuperscript{b}, Trapti Kalra\textsuperscript{a}

\textsuperscript{a}Employee at IBM
\textsuperscript{b}Intern at IBM

Abstract

In the past few years, texture-based classification problems have proven their significance in many domains, from industrial inspection to health-related applications. New techniques and CNN-based architectures have been developed in recent years to solve texture-based classification problems. The limitation of these approaches is that none of them claims to be the best suited for all types of textures. Each technique has its advantage over a specific texture type. To address this issue, we are proposing a framework that combines existing techniques to extract texture features and displays better results than the present ones. The proposed framework works well on the most of the texture types, and in this framework, new techniques can also be added to achieve better results than existing ones. We are also presenting the SOTA results on FMD and KTH datasets by combining three existing techniques, using the proposed framework.

1. Introduction

Texture analysis of images plays a vital role across multiple businesses. Image texture has numerous definitions according to several authors and every one of them defines unique ways to calculate discrete qualities of texture like smoothness, roughness, uniformity, homogeneity, etc. All of them are based on the concept of repeating patterns in an image Liu and Aldrich (2022); Tuceryan and Jain (1993). Texture analysis is used by many companies including medical imaging Liu et al. (2019), remote sensing Zhai et al. (2021), industrial inspection of materials Silven (2000), autonomous vehicles Zu et al. (2015), explosive hazard detection Petersson and Gustafsson (2016) and in every other industry. Even though there are diverse uses of texture analysis, one of its main components relies on texture classification. The spatial distribution of pixels from an image is an integral part of texture classification Julesz (1981). Convolutional Neural Network (CNN) has been very successful in capturing the local and global spatial features, which play a key role in texture classification. CNN preserves the relative spatial information with the help of convolution layers and aggregates the spatial information using the pooling layers Xue et al. (2018). Local spatial features assist in recognizing the patterns of a texture. In an image, while classifying the texture, some local patterns are repeated throughout the image Tuceryan and Jain (1993). These local features exhibit almost the same characteristics as one another and they are extremely critical in differentiating the textures. The traditional CNN architectures do not exhibit phenomenal results for classifying the textures merely based on the spatial features. The CNN can separate the local features with the aid of convolution layers although these spatial features are aggregated by employing the pooling layer. They make decisions primarily based on the global features rather than focusing on the local features. CNN tends to lose the locally rich features and that’s the major reason behind the underperformance of CNN while identifying textures Chen et al. (2021). To overcome this problem, numerous techniques have been introduced where TE layers are applied before the fully connected (FC) layers, using which local features are extracted. They are used along with global features. Several techniques such as DEPNet Xue et al. (2018), DeepTEN Zhang et al. (2017), CLASSNet Chen et al. (2021), FENet Xu et al. (2021), etc., have been built based on this strategy. Each of these methodologies focuses on some crucial aspects of texture and is specialized for specific use-cases. Using a Histogram layer Peeples et al. (2021) along with CNN produces adequate results, whereas FENet Xu et al. (2021) produces the state-of-the-art (SOTA) results for some benchmark texture datasets. In this paper, we propose a method where we concentrate on combining several techniques rather than developing a new one. When various TE processes are combined, we can effortlessly capture the divergent characteristics of a texture in a better manner as each of these processes emphasizes unique aspects of texture. We achieved SOTA results on the major texture datasets by integrating distinct existing approaches i.e, DeepTEN, Histogram, and FAP Zhang et al. (2017); Peeples et al. (2021); Xu et al. (2021). During the implementation of this strategy, we have a greater superiority over each method individually, since we classify texture based on distinct characteristics rather than focusing on one major trait of texture.

2. Contribution

In this paper firstly, we explored and proved a hypothesis that a group of distinct and unique weak learners when ensembled together form a strong learner. To prove this hypothesis we have suggested a framework where more than one TE technique has
been ensembled after the convolution layers. In general, the TE techniques use two methods during ensembling. However, our method proposes a framework that enables us to ensemble more than two techniques at the same time. For feature aggregation, previous researchers used bi-linear pooling yet we apply concatenation - in a view of the fact that it reduces the number of parameters due to less activation size before passing it to the FC layer. Secondly, we prove that orderless pooling has very less impact on the model when it is concatenated with another texture feature extraction technique. Whilst we are concatenating distinct techniques post convolution layer, we can replace the orderless pooling layer with some other mechanism considering the point it makes a minimal impact on the accuracy of the model. Benefitting from the proposed framework we have effectively ensembled numerous feature extraction techniques to achieve state-of-the-art (SOTA) results on bench-marked texture-based datasets.

3. Motivation and main idea

Whenever texture classification is performed, there is no perfect method that works flawlessly for all texture-based datasets. Combining multiple techniques can provide a finer solution to a given dataset. Inspired by the idea of passing various related inputs rather than single input to yield better results, based on the research work by Pandey et al. Pandey and Jha (2020). We employ this idea to pass additional informative features to determine better results. We developed an idea of ensembling varied techniques to be applied on top of the backbone to build a model which is more efficient. This ensembling technique works by classifying texture based on the variety of texture features, which are acquired using individual techniques. Our experiments demonstrate that merging the key features/advantages of dissimilar techniques improves the performance of the produced model as compared to the individual methods. Since texture is the key feature in many image classification tasks, therefore, texture-based classification is one such concept that assists in various image classification tasks and is used in many real-life applications. Because of its wide range of use cases and provided that impeccably advanced mechanisms can be employed in the texture-based classification task, a flexible and scalable model is required which can be used for a variety of texture-based datasets. As the datasets can have many classes which share similar characteristics to some extent, we aim to construct a scalable and flexible model. From the previously developed techniques namely DeepTEN: Texture Encoding Network Zhang et al. (2017), DEPNet Xue et al. (2018), DSRNet Zhai et al. (2020), etc. We can say that applying the texture feature extraction techniques after the activation layer of a backbone results in a better accuracy compared to techniques where the backbone is not used. Once the input image is passed through the backbone i.e. pre-trained weights of the convolution and pooling layers of the backbone model, subsequently local features, as well as the global features, can be examined to extract texture efficiently to classify the texture with improved preciseness. Numerous techniques have been developed previously to solve plenty of problems. Various techniques can be used to partially satisfy the goal of a project, yet there are very rare chances of finding a model/technique that completely suits the same goal as one’s project. Applying several techniques to a certain dataset may generate bad results, this can be avoided if they are combined mindfully. Which would yield exemplary results. A group of weak learners when combined via ensemble methods produce a strong learner. To produce an efficient ensemble model, we need to choose our methods astutely, as we don’t want to combine two techniques that are competent at doing the same task. Let’s say we have a model which uses technique $A$ to distinguish between dog and cat from the shape of ears and another model using technique $B$ to discriminate between dog and cat from the color and size of ears. If we combine techniques $A$ and $B$ there is a minor/negligible chance of improvement in the model. Rather if we use a technique $C$ which recognizes dogs and cats based on the shape of the body. Then combining technique $C$, along with technique $A$ or $B$ shall exhibit significant improvement in the result. The obtained accuracy shall be greater than the individual techniques. We use this idea to create an ensemble model which will save the effort of creating new techniques whenever there is a need for improvement in the accuracy. We need to select techniques in such a way that there is less correlation between the techniques to maximize the utilization of each technique while providing the best possible output. According to our idea, each technique which is chosen for the ensemble model should work on separate aspects of the data. This will ensure the smooth working of the model and will ensure that the resultant model is better than individual models. Using any specific technique for TE limits its use-case for a specific variety of textures. For other texture types that technique may not work. If one can use various methods in a single architecture then it would be able to handle complex texture types.

4. Related Work

GAP has been used by various people for feature aggregation TCNN Andrearczyk and Whelan (2016) and wavelet CNN Fujieda et al. (2017) are some of the most famous examples of the application of GAP. It is very lossy regarding spatial information but is very suitable for extracting the texture features Fujieda et al. (2017). The bi-linear pooling (BLP) introduced by Lin et al. Lin and Maji (2016) is used to make second-order interactions between the two outputs Dai et al. (2017), it is an outer product of each pair of feature points. BLP encodes more information than GAP GAP also acts as a structural regularizer which enables us to explicitly enforce feature maps to classify the textures more efficiently Lin et al. (2013). DeepTEN Zhang et al. (2017) is one such technique that uses residual dictionary learning to encode features on an end-to-end learning framework. It is different from SIFT Lowe (2004) and filter banks Cimpoi et al. (2015) as hand-engineered features are used, but in DeepTEN dictionaries, features, classifiers, and encoding representation are all trained together to produce an orderless encoding. DeepTEN is very efficient for material and
texture recognition tasks, meanwhile, it also has a unique ability to be used as a pooling layer on top of convolution layers as it increases the flexibility of the deep model Liu et al. (2019). To integrate histograms with CNN a histogram layer was introduced by Joshua Peeples et al. (2021), which captures the local spatial features using a histogram. The histogram layer uses a Radial basis function (RBF) Bors (2001) for the binning operation to find the bin centers and widths. It also makes use of the GAP to capture the spatial, texture, and orderless convolution features. The histogram layer enables the model to use fewer layers rather than conventional CNN, as the texture information is directly captured by feature maps Peeples et al. (2021).

Fractal analysis pooling (FAP) is used in the FENet to group the points of a texture based on the local fractal dimension of the image. It eliminates the dependency on spatial order while encoding the characteristics of features Xu et al. (2021). Fractal dimension is a quantitative measure of the roughness of a given image Shanmugavadivu and Sivakumar (2012). GAP is used to capture the spatial features of a texture but sometimes it fails to distinguish between complex distributive patterns from a texture. While FENet uses both GAP and FAP as the method and distributive features are captured from distinct aspects Xu et al. (2021). As shown by Pandey et al. Pandey and Jha (2020) the approach of incorporating more diverse inputs into a single model shall demonstrate a significant improvement in results. We propose a model in a similar guideline where the proposed architecture shall exhibit an improvement in results by combining several methods and techniques after the convolution layers. In our ensemble methods, we have considered only those methods which are used to extract from a single feature map unlike the class-net Chen et al. (2021) where numerous feature maps are used to find the texture feature.

5. Ensemble of texture extraction technique

5.1. Limitation of bi-linear pooling

Bi-linear pooling (BLP) has been utilized to make the second-order interaction between the two outputs. A drawback of using second-order interaction is we can only use two outputs to participate in bi-linear pooling. This concept stops us from using more than two outputs. The output of the bi-linear pooling layer has a lot of parameters when compared to an alternative method of concatenation. Let’s say there we have two activations with sizes of m and n; as shown in Figure 1 while doing the bi-linear pooling between two activations each feature of input 1 is pooled with each feature of input 2 yielding a size of \( m \times n \). Meanwhile in Figure 2 while concatenation is done all the features of input 1 and input 2 are simply joined, which yields a size of \( m + n \). From this, we can simply say that while combining two techniques the number of parameters involved in BLP is greater than or equal to concatenation i.e. \( m \times n \geq m + n \). Apart from the number of parameters involved the concatenation can deal with more than two techniques at the same time. For the above-stated reasons, we use concatenation rather than BLP. However, we can apply BLP for more than two inputs, but it would not be explainable to make second-order interaction between two inputs while leaving other inputs as it is. Bi-linear pooling is not a suitable framework where several inputs can be used to extract features.

5.2. Idea for selecting texture extraction techniques

While selecting a TE technique one should keep in mind that the selected technique is diverse. Ensembling a technique that works exceptionally on dataset \( A \) with another technique that works adeptly with dataset \( B \) makes the method more flexible and diverse. This new method should produce up to the mark results as long as the experiments are conducted using the same backbone. Two distinct strategies can only be compared when they operate on the same backbone. During the selection of different TE techniques, we need to ensure that the difference between the sizes of activation output should not be exceptionally large. It would impact the performance of the technique with smaller activation output. For instance, merging two techniques \( A \) with an activation output size of 512 and \( B \) with an activation output size of 48 will exert more importance on technique \( A \). Pointwise convolution technique can be used to reduce the activation output size.
5.3. Selected techniques and their hyperparameter details

5.3.1. Histogram

The histogram layer models feature maps using RBFs Bors (2001). The dimension of these feature maps is two whenever image data is used. Whereas the dimension is one for time series/signal data and it is three for spatial or volumetric data. The standard histogram operation counts the number of values that fall within certain ranges is called as “number of bins”. The center of these ranges is defined as “bin centers” and the interval or size of each range (or “bin”) is defined by the “bin width”. The input size for the convolution layer is calculated by the formula,

\[
\text{feature_map_size} = \frac{\text{feature_map_size} \times \text{number_of_bins}}{	ext{number_of_bins}}
\]

Which would be \(2048/(4*4) = 128\) for ResNet50 and \(512/(4*4) = 32\) for ResNet18. The output size of the convolution layer is calculated as a product of the number of bins and the input size. In our case it is \(4*128 = 512\) for ResNet50 and \(4*32=128\) for ResNet18. Meanwhile, the bin width for the input and output would remain the same, i.e., \(512\) for ResNet50 and \(128\) for ResNet18. Finally, for better compatibility of the model grouping is important which is \(512\) groups.

5.3.2. DEEPTEN

DeepTEN is a framework where the feature extraction, dictionary learning, and encoding representation are learned together in a single network. For this framework, the encoding layer accepts features or feature channels of dimension 512. The Encoding Layer is a layer where the dictionary learning and residual encoding happen in a single layer of CNN. The Encoding Layer learns an inherent Dictionary. The Residuals are calculated by the pairwise difference between the input visual descriptors and the codewords of the dictionary. The number of codewords \(n_{codes}\) used in our case is 8. Finally, the residual vectors are aggregated with the assignment weights. Next is the linear layer with input features as \(512*n_{codes}\) which is 4096 and output features as 128. Lastly, batch normalization is done with 128 features.

5.3.3. Fractal

FE module, which leverages multi-fractal geometry and uses a hierarchical fractal analysis process for encoding the regularity of spatial arrangement in a feature map. The FE module is implemented in four sub-processes whose main objective is to calculate the fractal geometry. Before the first sub-process, a convolution layer is required for the effectiveness of the FE module. The parameter for this Conv2d layer with 512 inputs and outputs is a kernel of size 3 and Batch Normalization with 3 features over a 4D input. Lastly, a ReLU activation function is applied with default values. The first sub-process is FAP which first compresses the input feature tensor to \(C_0\) channels by a \(\text{Conv1x1} \rightarrow \text{BN} \rightarrow \text{ReLU}\) block. FAP processes the new feature tensor lice by the slice and then concatenates the results in overall slices. The number of features \((K)\) is 16; which is the same as \(C\) from the expected input size \((N, C, H, W)\). The next block is the Local Dimension Estimation Block (LDEB) which calculates a point-wise local fractal dimension map \(D\) from the input feature slice \(X\). The main parameter in this process is the local fractal dimension map \((D)\) which is 1 in our case. This is used as input channels for the Conv2d layer. There are six conv2d layers in this fractal module. The output of LDEB is a feature slice which is an input for the Point Grouping block (PGB). The PGB categorizes the entries on the feature slice based on the map \(D\) where \(D\) is the Local Fractal Dimension Map with a value of 1. Finally, the Global Dimension Calculation Block (GDCB) takes the map \(D\) and adds max-pooling layers which are 6 in our case. The same as the convolution layers in the LDEB block. The kernel sizes used are 1, 3, 5, 7, 9, and 11 in the 6 max-pool layers respectively. The output of the GDCB block is the dimension of the fractal to be calculated.

5.3.4. GAP

Global Average Pooling (GAP) is a pooling operation designed to replace FC layers in classical CNN. The idea is to generate one feature map for each corresponding category of the classification task in the last MLP convolution layer. One advantage of global average pooling over the FC layers is—it is more native to the convolution structure by enforcing correspondences between feature maps and categories. The hyperparameters that are used in the process is an average pooling, which applies a \(2D\) average pool over an input signal composed of several input planes. The size of the window or better known as kernel size used here is 7. CNN commonly uses convolutional kernels with odd height and width values such as 1, 3, 5, and 7. Choosing odd kernel sizes has the benefit, we can preserve the spatial dimensionality while padding with the same number of rows on top and bottom, and the same number of columns on left and right. Next to apply a linear transformation to the incoming data the hyperparameter utilized is a linear layer with input features as 512 and output features as 48. Next, we apply a batch normalization over the input with the number of features as 48.

5.4. Ensembling texture extraction techniques produces good result

To produce substantial outcomes while doing any classification tasks, it’s not always necessary to have the best feature extraction method to do the task. It can also be done by ensembling a group of feature extraction tactics that have unique working principles. Provided these methods are chosen wisely, they may reflect superior accuracy to the best existing method. A group of weak learners tends to function more efficiently or at least equally in comparison to a single strong learner. However, as mentioned earlier the selected methods need to be unique in their working principles and preferably they shouldn’t have any overlapping mechanisms among them. Failing on this may increase the computation process as well as may result in less accuracy as the model might be confused because of their sim-
Figure 3: Framework for combining more than one technique

As seen in Table 1, when GAP is precluded while ensembling the other techniques then accuracy has increased by over 1 percent, this is a significant growth in the accuracy. This could be because the GAP and histogram are both dependent on similar aspects of the spatial distribution of texels. When there is no such overlap/correlation between the techniques the model gives a better result. This proves that while ensembling the feature extractors their combination must be on the mark, even though they may produce good results there is always a scope for improvement with some other unique techniques.

| Dataset | With GAP | Without GAP |
|---------|----------|-------------|
| FMD     | 83.2     | 84.3        |
| KTH     | 87.7     | 89.0        |

Table 1: Comparison between techniques with and without GAP

5.5. Reason for including discussed 4 techniques while leaving others

We have chosen the techniques of DeepTEN Zhang et al. (2017), histogram layer Peeples et al. (2021), FAP Xu et al. (2021), and GAP while designing our proposed model. Other methods such as DEPNet Xue et al. (2018), DSRNet Zhai et al. (2020) and CLASSNet Chen et al. (2021) are just inherited from one of the used methods. Using them would cause an overlap while capturing the characteristics of the texture. Using other methodologies along with these four techniques will make the model unnecessarily complicated as two strategies will be working similarly to determine textures from a dataset. DEPNet and DSRNet use the same encoding layer that has been used in DeepTEN. DEPNet has been developed so that the texture can be better recognized for the images in the wild. It utilizes the same encoding layer along with a GAP layer which empowers the model to handle both orderless components and ordered spatial information. DSRNet Xue et al. (2018) which was developed for ground terrain recognition also uses the same texture encoding layer from DeepTEN. DSRNet is a model inspired by DEP and it perceives spatial dependency among multiple texture primitives and leverages it as inherent structural property for texture recognition. DSRNet has been built using ResNet50 He et al. (2016) as a backbone which also makes it laborious to
integrate/ensemble with other mechanisms Zhai et al. (2020). Whereas, CLASSNet uses a concept of statistical self-similarity (SSS) Mandelbrot (1967) which is calculated with the help of fractals which is something similar to FAP. In CLASSNet the feature maps are learned using CNN and have a cross-layer SSS for exploiting the SSS of images. For the SSS-based texture recognition, fractal analysis is used to characterize the SSS on images Chen et al. (2021). The fractal analysis is also used in FAP for calculating the fractal dimension and classifying the texture based on the same. Considering the other methods as just an extension of the already used methods, we proceed to use the parent method since we are ensembling the distinct techniques and we don’t want the texture to be categorized based on the same attributes umpteen times.

6. Comparison with other texture extraction techniques

Our TE technique mainly concentrates on getting hold of various texture features with very less correlation among each other. We want to capture and use the peculiar characteristics of an image that define texture and are useful for classifying textures into their appropriate classes. GAP is a pooling layer but also has the capability to be used in many techniques Fujieda et al. (2017); Peeples et al. (2021); Xu et al. (2021) to capture local-to-global features. When BLP is used to combine distinct inputs from individual branches of the convolution neural network then we will need numerous BLP layers as only two inputs can participate at once. Another drawback of using BLP is the output size of the activation increases as the product of both the inputs.

While an underrated method of combining multiple inputs is concatenation, we can retain the information from the previous layer in the same format before and after concatenation. When there is minimal correlation between the characteristics extracted we can not afford to lose any kind of information extracted by each of these techniques. For this reason concatenation is used to combine the DeepTen Zhang et al. (2017), Histogram layer Peeples et al. (2021) and FAP Xu et al. (2021) to produce SOTA results for KTH Caputo et al. (2005) and FMD Sharan et al. (2009) datasets. DeepTEN is one such method where feature extraction, dictionary learning, and encoding representation all happen at the same time. The learned convolution features are easily transferred since the encoding layer learns an inherent dictionary that carries domain-specific information. DeepTEN is a flexible framework that allows arbitrary input image size making it easier to combine with any model. At the same time, FAP focuses on discriminating textures based on the fractal dimension i.e., based on the roughness of the texture. As mentioned earlier, histogram layer captures the texture information directly from the feature maps and is based on the fundamentals of local histograms which can be used to distinguish textures. All these techniques emphasize completely contrasting characteristics of the texture to classify them into separate classes and ensembling these unique techniques assists us to do the texture classification task more elegantly. When various characteristics are taken into consideration then our model has more information and can easily draw a boundary between distinct texture classes.

7. Experiment evaluation

7.1. Experiment setup

After passing the input through numerous convolution layers, we combine the technique of histogram layer along with texture encoding and fractal analysis pooling layers. Using this concept, we can extract texture features from a single feature map. To check the effectiveness of our model we ran this model on 2 benchmark datasets KTH and FMD. For KTH we used the random 10 splits and for FMD recommended split size is used. The mean and standard deviation for the results are calculated for all the splits and presented in the results table. ResNet-18 is used as a backbone and the model is trained using cross-entropy loss via an SGD optimizer along with a cosine scheduler. The model is trained for 40 epochs for KTH and 60 epochs for FMD. We have set the batch size of 32 for KTH and 16 for FMD. Meanwhile, the learning rate has been set to $1e^{-2}$ on KTH and $1e^{-3}$ on FMD. For better stability and optimization of the model, the training dataset is transformed by random horizontal flipping and random cropping on an image of size 256 x 256. The model has been built using Pytorch 1.7.1 version and all the experiments presented in this paper have been run on NVIDIA Tesla K80.

7.2. Results

Our model is compared to numerous deep learning models for texture classification like LSCNet, MAPNet, DSRNet, CLASSNet, and FENet. The results for the same are presented in the table below (Table 2). The results written in the table are directly taken from the existing papers and left blank wherever the result is not available. Our method gives the best accuracy for KTH and FMD, these results are for the ResNet18 backbone. In comparison to the other methods, our model has increased the accuracy on benchmark datasets by approximately 2 percent which is a significant improvement in terms of accuracy.

| Method    | FMD          | KTH          |
|-----------|--------------|--------------|
| CLASSNET  | 82.5±0.7     | 85.4±1.1     |
| DeepTEN   | 80.2±0.9     | 84.5±3.5     |
| DSRNet    | 81.3±0.8     | 81.8±1.6     |
| MAPNet    | 80.8±1.0     | 80.9±1.8     |
| LSCNET    | 76.3±0.1     |              |
| FENET     | 82.3±0.3     | 86.6±0.1     |
| Ours      | 84.3±1.1     | 89.0±2.34    |

Table 2: Comparison with SOTA

For further analysis of the ensemble methods, we further experiment by ensembling three techniques out of the four techniques
and also we do the same by experimenting with two techniques at the same time. The results for ensembling three techniques at the same time can be seen in Table 3, and in Table 4 the results for the two techniques have been compiled. All of these results are computed for the FMD dataset.

| Dataset | DeepTEN | GAP | Histogram | FAP | Accuracy |
|---------|---------|-----|-----------|-----|----------|
| FMD     | No      | Yes | Yes       | Yes | 84.3     |
|         | Yes     | No  | Yes       | Yes | 81.9     |
|         | Yes     | No  | No        | Yes | 83.1     |
|         | Yes     | Yes | Yes       | No  | 83.4     |
| KTH     | No      | Yes | Yes       | Yes | 89.0     |
|         | Yes     | No  | Yes       | Yes | 87.4     |
|         | Yes     | Yes | No        | Yes | 88.1     |
|         | Yes     | Yes | Yes       | No  | 88.8     |

Table 3: Comparing three techniques at once

| Dataset | DeepTEN | GAP | Histogram | FAP | Accuracy |
|---------|---------|-----|-----------|-----|----------|
| FMD     | Yes     | Yes | No        | No  | 81.9     |
|         | Yes     | No  | Yes       | No  | 81.4     |
|         | Yes     | No  | No        | Yes | 78.6     |
|         | No      | No  | Yes       | Yes | 54.0     |
|         | No      | Yes | Yes       | No  | 83.11    |
|         | No      | Yes | No        | Yes | 83.5     |

Table 4: Comparing two techniques at once

In Table 5, we also demonstrate how each technique uniquely performs on the FMD dataset.

| Method   | Accuracy |
|----------|----------|
| DeepTEN  | 83.5     |
| GAP      | 81.1     |
| Histogram| 72.9     |
| FAP      | 33.8     |

Table 5: Each technique on FMD

7.3. Effectiveness of proposed ensemble technique

The proposed technique outperforms the various models which were put forward on this topic before. It is all related to the various mathematical problems of time and work i.e., if A can do a piece of work in 10 days and B can do a piece of work in 15 days then how much time will it take if both of them do the work simultaneously. Similarly if a model A is performing well on a specific dataset with a certain backbone but has some drawback which model B can fulfill. Then combining both of them will ensure superior results as long as both have the same backbone. Our technique works on the same principle of combining techniques that are diverse to produce a better result than the single working model. The various combinations were looked upon and experimented such as combining GAP with DeepTEN, FAP with Histogram, GAP with FAP and DeepTEN, and all four of them combined. It turns out that the combination of DeepTEN, FAP, and Histogram was performing extremely well on KTH and FMD dataset resulting in SOTA. We fabricated more than a 2 percent jump in the accuracy of both datasets in comparison with earlier methodologies. The more general way to look at this framework is to combine as many effective methods as possible for the classification. However, there is always an exception to every rule or approach. When all the four strategies were integrated, we observed a dip in the accuracy as contrasted to a combination of three methods. This would mainly happen due to over prediction and ambiguity in the models themselves. Suppose Model A individually gave the correct prediction on a certain image and Model B individually gave the wrong prediction on the same image so when we combine both of them the architecture changes and the weights also change resulting in either correct prediction or a wrong prediction. So the combination of methods needs to be properly investigated before using them.

7.4. More analysis

The four techniques that are taken into consideration are FAP, GAP, DeepTEN, and Histogram. We tried experimenting with various combinations of these four methods such as first considering only 3 out of 4, then 2 out of 4, and then lastly 1 out of 4, all of them adding to 15 experiments. For FMD we tried all these combinations while for KTH we tried only 3 out of 4 combinations. As shown in Table 3 and 4 the highest accuracy obtained on FMD-related experiments is 84.3 i.e., by combining the three techniques which are DeepTEN, Histogram, and FAP. The peak accuracy reached in the combination of two is 83.9 which had GAP and DeepTEN, whereas, the peak accuracy in the single combination is 83.5 which had only DeepTEN. So far, we can observe that DeepTEN is playing a major role in attaining the highest accuracy in every combination. In the three out of four combinations experiment, if we remove DeepTEN the accuracy dropped to 54 which is clearly stating how crucial DeepTEN is in TE. Moreover, in the two out of four combinations, the second and third highest accuracy was obtained by combining DeepTEN with FAP (83.5) and DeepTEN with Histogram (83.11).

One more thing to observe is when Histogram is removed from the combination of DeepTEN, Histogram, and FAP the accuracy dropped by merely 0.8, while, when FAP is removed the accuracy dropped by 1.2 percent, therefore, FAP is also playing a major role when it is in the combination but not individually because it is giving the worst accuracy 33.8 percent when tested alone. FAP gives good results only when combined with any of the other three. GAP on the other hand is showing neutral
changes when combined with any of the other three, increasing with Histogram and DeepTEN separately, nonetheless, accuracy dropped when combined with FAP. In the trio combination, it still shows a minimal increase in accuracy considering FAP is not in the trio. We can conclude that, when GAP has been removed, the accuracy reached its highest value which is 84.3 on the contrary when it’s introduced in the combination of all features the accuracy dropped by 1.1 percent. We also attempted to find the feature importance of each of the four used techniques in deciding the accuracy of classification task on FMD dataset. For this, We considered all 15 experiments which have been conducted for FMD dataset by trying all possible combinations of out of 4 methods. To achieve the feature importance values, we formulated a regression problem where we assumed accuracy values as regression values and encoded the presence of any technique in given experiment as 1 while absence as 0. We implemented Random Forest Regression algorithm on the formulated problem and calculated the feature importance from the trained model. We obtained the Figure 5 by this procedure. It suggests GAP has higher feature importance. In our case, it communicates that adding GAP tends to reduce the accuracy, on the contrary, removing GAP increases the accuracy. Furthermore, DeepTEN has the second highest in feature importance value, which conveys considering our experiments that adding DeepTEN helps in increasing the accuracy while removing it reduces the accuracy values. Histogram and FENet have third highest and the least feature importance value respectively.

8. Conclusion

In summary, we proposed a novel approach to encounter the downside of existing numerous texture feature extraction techniques when used individually. We developed a framework that incorporates a few texture-related techniques simultaneously and displays its effectiveness on benchmark texture datasets. The suggested framework has not only resulted in a state-of-the-art result but also made us understand the impact of using dissimilar techniques in a combination. In this paper, ensemble methods have always produced an improved result when compared to the individual model performance. This ensemble architecture can easily be employed in standard CNNs as well as sophisticated models such as DeepTEN, FENet, etc. We combined DeepTEN, wherein feature extraction, dictionary learning, and encoding representation- all happen at the same time. DeepTEN is a flexible framework that allows arbitrary input image size making it easier to combine with any model. FAP which focuses on discriminating textures is based on the fractal dimension. The third component histogram layer captures the texture information directly from the feature maps and is based on the fundamentals of local histograms which can be used to distinguish textures. While we only consider two remarkable texture datasets such as KTH and FMD to achieve SOTA results in this work, this technique can be applied to more datasets such as DTD, GTOS, and GTOS-M to demonstrate similar SOTA results. Our future work would be to investigate which TE methods to choose while combining unique feature extractors in light of the dataset characteristics to impact the results positively.

References

Vincent Andrearczyk and Paul F Whelan. Using filter banks in convolutional neural networks for texture classification. Pattern Recognition Letters, 84: 63–69, 2016.

Adrian G Bors. Introduction of the radial basis function (rbf) networks. In Online symposium for electronics engineers, volume 1, pages 1–7. DSP Algorithms San Jose, CA, USA, 2001.

Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-specific material categorisation. In Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1, volume 2, page 1597—1604. IEEE, 2005.

Zhile Chen, Feng Li, Yuhui Quan, Yong Xu, and Hui Ji. Deep texture recognition via exploiting cross-layer statistical self-similarity. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5231–5240, 2021.

Mircea Cimpoe, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3828–3836, 2015.

Xiyao Dai, Joe Yue-Hei Ng, and Larry S Davis. Fason: First and second order information fusion network for texture recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7352–7360, 2017.

Shin Fujieda, Kohei Takayama, and Yoshiha Hachisuka. Wavelet convolutional neural networks for texture classification. arXiv preprint arXiv:1707.07394, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Bela Julesz. Textons, the elements of texture perception, and their interactions. Nature, 290(5802):91–97, 1981.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.

Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep texture representations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2791–2799, 2016.

Li Liu, Jie Chen, Paul Fieguth, Guoying Zhao, Ramana Chellappa, and Matti Pietikäinen. From bow to cnn: Two decades of texture representation for texture classification. International Journal of Computer Vision, 127(1):74–109, 2019.

Xiu Liu and Chris Aldrich. Deep learning approaches to image texture analysis in material processing. Metals, 12(2):355, 2022.

David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004.

Benor Mandelbrot. How long is the coast of britain? statistical self-similarity and fractional dimension. science, 156(3775):636–638, 1967.

Vijay Pandey and Shashi Bhushan Jha. Incorporating image gradients as secondary input associated with input image to improve the performance of the cnn model. arXiv preprint arXiv:2006.04570, 2020.

Joshua Peeples, Weihuang Xu, and Alina Zare. Histogram layers for texture analysis. IEEE Transactions on Artificial Intelligence, 2021.

Henrik Pettersson and David Gustafsson. Multi-spectral texture analysis for ied detection. In Electro-Optical Remote Sensing X, volume 9988, pages 206–219, SPIE, 2016.

P Shannugadavudu and V Sivakumar. Fractal dimension based texture analysis of digital images. Procedia Engineering, 38:2981–2986, 2012.

Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Material perception: What can we see in a brief glance? Journal of Vision, 9(8):784–784, 2009.

O Silven. Applying texture analysis to industrial inspection. In Texture Analysis in Machine Vision, pages 207–217. World Scientific, 2000.

Mihran Tuceryan and Anil K Jain. Texture analysis. Handbook of pattern recognition and computer vision, pages 235–276, 1993.

Yong Xu, Feng Li, Zhile Chen, Jinxiu Liang, and Yuhui Quan. Encoding spatial distribution of convolutional features for texture representation. Advances in Neural Information Processing Systems, 34, 2021.

Jia Xue, Hang Zhang, and Kristin Dana. Deep texture manifold for ground terrain recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 558–567, 2018.

Wei Zhai, Yang Cao, Zheng-Jun Zha, HaiYong Xie, and Feng Wu. Deep
structure-revealed network for texture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11010–11019, 2020.

Hang Zhang, Jia Xue, and Kristin Dana. Deep ten: Texture encoding network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 708–717, 2017.

Zhaozi Zu, Yingjuan Hou, Dixiao Cui, and Jianru Xue. Real-time road detection with image texture analysis-based vanishing point estimation. In *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 454–457. IEEE, 2015.