Applied Computing and Intelligence

https://www.aimspress.com/journal/aci

Applied Computing and Intelligence
5(1): 112–126.
DOI: 10.3934/aci.2025008
Received: 10 December 2024
Revised: 04 April 2025
Accepted: 09 April 2025
Published: 16 April 2025

Research article

Exploring self-supervised feature extraction techniques in the K-views algorithm for texture analysis

Burak Kure[1], Min Wang[2], Coskun Cetinkaya[1] and Chih-Cheng Hung[1,*]

[1] Laboratory of Machine Vision and Security Research, College of Computing and Software Engineering, Kennesaw State University, Marietta, GA, USA
[2] Department of Mathematics, Kennesaw State University, Kennesaw, GA, USA

* **Correspondence**: Email: chung1@kennesaw.edu.

Academic Editor: Pasi Franti

Abstract: This paper explored self-supervised feature extraction in the K-views algorithm for texture analysis, which can reduce the reliance on labeled data. The study compared the supervised, and self-supervised methods, focusing on pair-wise feature comparison to assess the similarity and dissimilarity between texture classes. Among the techniques evaluated, Siamese networks achieved the highest performance, while self-supervised methods like rotation prediction demonstrated competitive results and scalability. Experimental results on KTH-TIPS, Kylberg, and UIUC datasets show the potential of self-supervised approaches in improving texture classification by leveraging feature representations without the need for manually labeled data for the K-views algorithm.

Keywords: K-views algorithm; texture analysis; image processing; machine learning; deep learning; Siamese networks; self-supervised learning

1. Introduction

In digital imagery, image texture analysis plays a pivotal role in computer vision, remote sensing, medical imaging, and industrial quality control [1,2]. Accurate texture analysis is vital for tasks like object recognition, segmentation, classification, and applications such as image retrieval, pattern recognition, and anomaly detection [3,4]. The K-views algorithm, a well-established method for image texture analysis [5], has evolved to address the complexities of modern image processing tasks.

Building on the preliminary work that introduced deep learning-based patch-wise feature extraction using Siamese networks within the K-views algorithm, this paper explores additional self-supervised feature extraction methods to improve performance while reducing the reliance on labeled data. Specifically, we propose a new connection prediction task that learns spatial relationships between patches to enable the K-views algorithm to capture structural features and spatial dependencies. This contribution represents a significant step forward in improving the effectiveness and versatility of self-supervised feature extraction methods.

Previous work incorporated deep learning features through Siamese networks, replacing the traditional use of Gabor filters on entire images, thus providing more granular texture analysis. This study evaluates self-supervised methods such as image inpainting [6] and rotation prediction [7], along with traditional methods like gray-level co-occurrence matrices (GLCM) [8], local binary patterns (LBP) [9], and scale-invariant feature transform (SIFT) [10]. We also assess supervised techniques, including convolutional neural networks (CNN) [11] and Siamese networks [12], for comparison.

While the introduction of self-supervised tasks such as connection prediction allows for the extraction of meaningful feature representations without requiring external labeled datasets, these methods have the potential for robust texture classification in data-scarce environments. The proposed connection prediction task, alongside methods like image inpainting and rotation prediction, introduces pretext tasks that generate supervisory signals directly from the data, enabling models to learn representations tailored for diverse and challenging datasets.

While Siamese networks provide the best results in terms of texture classification accuracy, the self-supervised methods also show significant promise, offering advantages like reduced reliance on labeled data and enhanced scalability. Methods such as image inpainting and rotation prediction contribute to robust feature extraction and improved classification performance by capturing rich texture representations. Furthermore, self-supervised feature extraction for the K-views algorithm holds promise for a combined approach, where the K-views algorithm can be trained using self-supervised methods without labeled data, followed by fine-tuning with a small labeled dataset.

The evaluation is conducted on three benchmark datasets, KTH-TIPS [13], Kylberg [14], and UIUC [15], which span various texture complexities and real-world applications. This study demonstrates the applicability of both traditional pattern recognition methods and self-supervised approaches which are the contributions to improve texture classification accuracy and the possibility of wide usage of the K-views algorithm.

The rest of the paper is organized as follows: Section 2 provides an overview of image texture analysis and the feature extraction methods used for comparison. The K-views algorithm is explained in Section 3. Section 4 explains the proposed feature extraction methods and their training, and integration with the K-views algorithm. Section 5 introduces the datasets and presents experimental results, while Section 6 gives the conclusion.

2. Image texture analysis

This section first reviews foundational and contemporary approaches to texture analysis, highlighting the shift from traditional handcrafted features to deep learning-based representations. It then introduces the feature extraction methods incorporated in this study, which are integrated into the K-views algorithm to explore their effectiveness in the context of texture analysis.

2.1. Related work in texture analysis

Image texture analysis is an essential technique in object recognition [16], image retrieval [17], image segmentation, and image compression. Texture, defined by the spatial arrangement of pixel intensities, encompasses various categories including random, repetitive, and irregular patterns [18]. Extracting meaningful texture features is indispensable for accurately interpreting and classifying image textures.

Traditional approaches to texture analysis have relied on handcrafted feature extraction methods. Among these, GLCM [8], LBP [9], and Gabor filters [17] have been widely used. The GLCM method analyzes the statistical distribution of pixel intensity pairs within an image, providing a compact representation of texture information [19]. LBP, in contrast, encodes local texture patterns by comparing pixel intensities within a neighborhood, offering simplicity and computational efficiency [9]. Gabor filters, inspired by the human visual system, are effective in capturing multi-scale texture features through spatial frequency analysis [17].

Similarly, the K-views algorithm employs a patch-based approach to explore texture characteristics by investigating a small area in which all pixels are expected to be similar for a class of texture [5]. Despite their effectiveness, these methods often struggle with capturing complex and hierarchical relationships in images, especially under varying conditions such as scale, illumination, or noise.

In recent years, deep learning has emerged as a transformative force in image texture analysis. The CNN has demonstrated exceptional capabilities in learning hierarchical texture representations directly from raw image data. These models excel at capturing complex inter-pixel relationships, making them well-suited for texture classification and segmentation tasks [20,21].

Self-supervised approaches, such as those based on image inpainting [6] and rotation prediction [7], have further extended the scope of texture analysis by leveraging unlabeled data to improve feature extraction. *Self-supervised learning* is a paradigm where models learn feature representations by solving auxiliary tasks, with labels generated automatically from the data itself. Unlike reinforcement learning, which focuses on learning policies through reward-based interaction with an environment, self-supervised learning focuses on designing pretext tasks to extract meaningful representations [22].

Hybrid methods combining traditional and deep learning-based techniques have also shown promise. For instance, traditional texture descriptors such as GLCM and LBP have been used to complement deep learning models, either as inputs or as feature pre-processors [23,24]. Conversely, CNN can be employed to generate rich feature representations, which are subsequently classified using traditional machine learning algorithms [25]. These hybrid approaches aim to combine the strengths of both paradigms, addressing the limitations of individual methods and enhancing performance across diverse datasets.

2.2. Feature extraction methods used in this study

The K-views algorithm [5] is systematically evaluated using a variety of feature extraction methods, each integrated as a distinct approach in the algorithm's feature extraction step. This evaluation aims to determine the strengths and limitations of each method in capturing texture features effectively.

The GLCM [8] is the first method evaluated for feature extraction in this study. By analyzing the spatial relationships among pixel intensity values, GLCM captures texture properties such as contrast, homogeneity, and entropy [19]. Feature vectors of 48 dimensions were generated by examining co-occurrences at specific distances ([1, 2, 3]) and angles ([0, π/4, π/2, 3π/4]). This method provided a balance between computational efficiency and the depth of representation, forming a benchmark for comparison.

The LBP is also evaluated, focusing on encoding microstructural texture patterns by comparing a central pixel to its neighbors [9]. This approach emphasizes capturing local variations in texture, complementing the global features extracted by GLCM, and offering an additional perspective for texture classification.

To address dimensionality challenges, principal component analysis (PCA) was incorporated as a feature extractor [26]. It reduces the feature space by projecting high-dimensional data onto a subspace defined by the top 20 principal components. This dimensionality reduction enhances the algorithm's computational efficiency while preserving critical texture-related information.

The SIFT provides an alternative approach by generating 128-dimensional feature vectors that are invariant to scale, rotation, and affine transformations [10]. This method proves particularly robust for texture datasets with variations in viewpoint and scale, making it a valuable addition to the evaluation.

Deep learning-based methods were extensively explored in this study. A CNN autoencoder [27] was trained to reconstruct texture patches, allowing the encoder to extract compressed, meaningful representations as shown in Figure 1. Similarly, a variational autoencoder (VAE) [28] was employed to capture high-level latent features, offering a more abstract representation of texture characteristics compared to traditional methods. Both autoencoder-based approaches demonstrated the potential of unsupervised learning in extracting deep, nuanced features for texture classification.

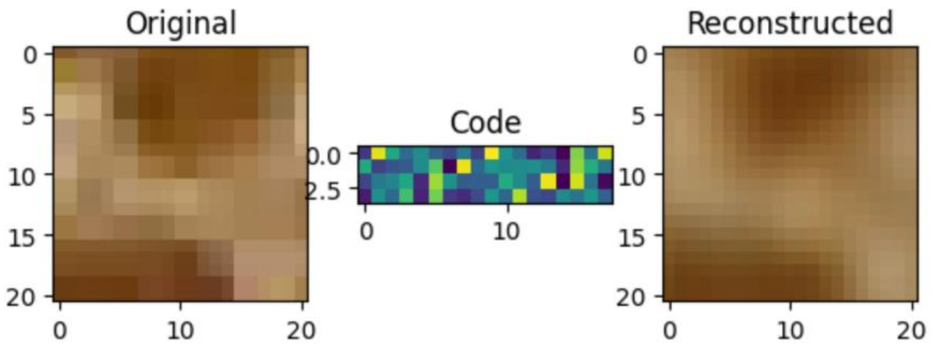

Figure 1. Example of an input patch, the extracted features, and the reconstructed output.

A CNN-based feature extractor trained specifically for texture classification is also evaluated. By training the CNN for classifying the textures and after the training using the penultimate layer of the network for feature extraction, high-level texture features are obtained, demonstrating the effectiveness of supervised learning in extracting discriminative patterns.

Lastly, Siamese networks [12] introduce another supervised learning dimension for the evaluation. These networks are trained on input pairs to predict whether they are similar or dissimilar. The Siamese neural network architecture as shown in Figure 2 is designed for similarity learning. It processes two input vectors through a shared convolutional neural network to obtain feature encodings.

These encodings are compared through a differencing layer that computes the Euclidean distance between the feature vectors. This distance is then transformed into a similarity score, typically using a sigmoid function to map it to a range of [0, 1], where a score near 1 indicates high similarity and a score near 0 indicates low similarity. The similarity score is subsequently compared to the true label using a loss function, such as binary cross-entropy, which quantifies the discrepancy between the predicted similarity and the actual label. After training, the output of the encoding layer can be utilized for feature extraction, which is particularly useful for clustering tasks.

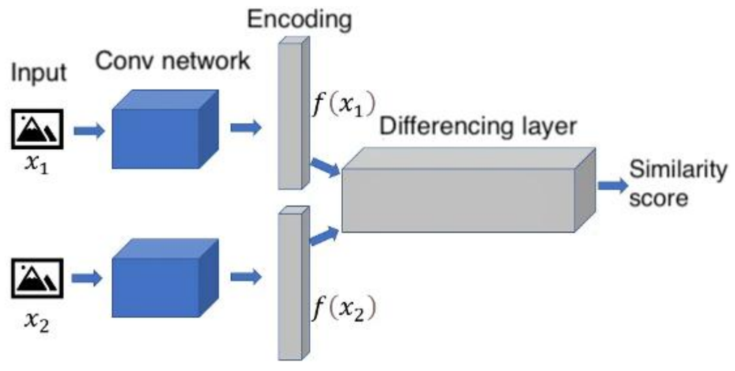

Figure 2. The architecture of Siamese networks for similarity learning [29].

Each of these feature extraction methods is independently assessed within the framework of the K-views algorithm, providing a diverse evaluation of traditional, unsupervised, and supervised approaches. This comprehensive exploration underscores the versatility of the algorithm and its adaptability to various feature extraction techniques, offering valuable insights into the strengths and limitations of each method for texture analysis.

3. The K-views algorithm

The concept of *characteristic views* has emerged as a useful tool in image texture classification, introduced in the K-views algorithm [5]. The research addressed the challenge of classifying textures by proposing an approach centered on the extraction of characteristic views in which distinctive features are derived from sample sub-images of each texture class. This innovation forms the foundation of the K-views algorithm, which operates on the premise that many natural textures exhibit repetitive patterns observable from different spatial perspectives.

The K-views algorithm begins by selecting random sub-images from each texture class as representative samples. From these sub-images, small blocks encapsulating texture features are extracted, forming the initial set of views. Clustering algorithms, such as K-means or fuzzy K-means, are then employed to group these views and identify a specified number, *K*, of characteristic views for each texture class. During classification, views from the input image are matched against these characteristic views, with each pixel being assigned to the class corresponding to the closest match.

Two key parameters significantly influence the performance of the K-views algorithm: the size of the views and the number of characteristic views (*K*). Larger view sizes and a higher *K* generally enhance classification accuracy by capturing more detailed texture information. However, this comes at the cost of increased computational requirements. Despite this trade-off, the K-views algorithm has demonstrated exceptional performance, offering a robust and flexible framework for texture classification and image analysis.

The approach's strength lies in its ability to model repetitive and spatially varying patterns in textures effectively. By characterizing representative views, the algorithm gives high accuracy, making it particularly suitable for applications where precise texture classification is critical. The basic K-views algorithm is sketched below (Algorithm 1):

Algorithm 1. The K-views algorithm for texture classification.

```
Input: Image $I$, number of characteristic views $K$, view size $Vs$, number of clusters (classes) $N$
Output: Classified texture class.
Steps:
1. Preprocessing the image $I$, by applying the Gabor filter.
2. Sample sub-images from each texture class in $I$.
3. For each sub-image:
   - Extract views (blocks) of size $Vs$.
   - Apply the clustering algorithm to derive $K$ characteristic views from these blocks.
4. For each view in the original image $I$:
   - Compare the view with the characteristic views across all classes.
   - Assign the class of the best matching characteristic view.
5. Classify the image based on the cluster with the highest count of classified patches, assigning the texture class corresponding to that cluster.
Return: Classified texture class.
```

4. The proposed feature extraction methods

The proposed feature extraction methods are to enhance the K-views algorithm for texture classification by incorporating advanced feature extraction techniques, focusing on self-supervised learning methods. Our contributions are summarized in the following:

- The novel *connection prediction* method, alongside existing self-supervised approaches like *image inpainting* and *rotation prediction*, is introduced. These methods allow feature extraction without requiring labeled data, making them highly versatile for diverse datasets.
- The use of self-supervised learning methods for feature extraction involves training on patch pairs generated directly from the data itself, with labels automatically derived based on the design of the pretext task. This approach eliminates the reliance on externally labeled datasets, as the required supervisory signals to train the feature extractors are inherently created from the training data.
- The method for generating these pairs varies depending on the specific self-supervised task, each designed to force the model to learn meaningful texture representations without requiring labeled data.

For image inpainting, random patches are corrupted by masking a region as illustrated in Figure 3. The task for the model is to reconstruct the original patch. Through this reconstruction process, the model learns a feature representation by focusing on understanding the surrounding context of textures, enabling it to capture complex texture relationships and details.

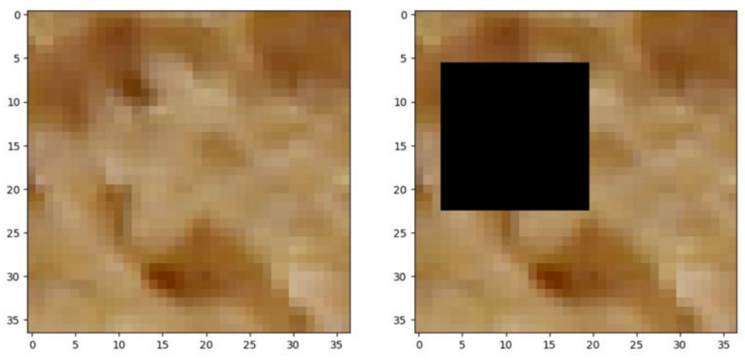

Figure 3. Example of an original patch (left) and masked patch (right) in image inpainting.

In the case of rotation prediction, a patch is rotated by a random angle chosen from $\{0°, 90°, 180°, 270°\}$ as shown in Figure 4, resulting in a transformed patch. The model's task is to predict the rotation applied, which helps it learn rotation-invariant features. This approach is particularly useful for texture analysis, as it ensures the model remains robust to variations in orientation.

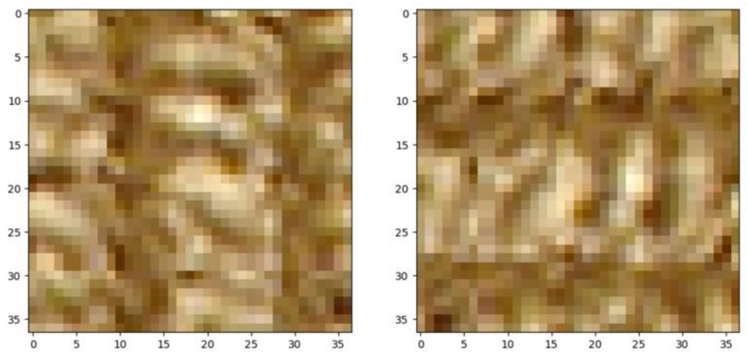

Figure 4. The original patch (left) and rotated patch by 90 degrees (right).

For the proposed connection prediction task, two patches are sampled either from neighboring or non-neighboring regions of the same image as demonstrated in Figure 5. A label is assigned to indicate whether the patches are spatially connected with 1 or not connected with 0. This task trains the model to learn spatial relationships between patches, enhancing its ability to capture structural features and spatial dependencies in texture patterns.

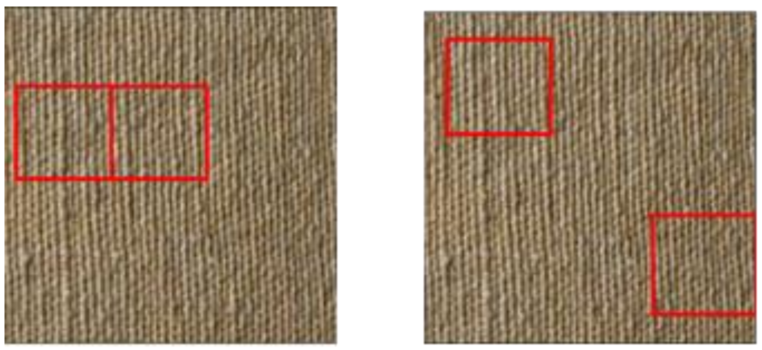

Figure 5. Selecting connected and unconnected pairs for training with connection prediction, respectively.

4.1. Training the feature extractors

Each self-supervised method employs a deep learning-based architecture to train feature extractors, tailored to the specific task of learning meaningful texture representations.

For image inpainting, a CNN-based encoder-decoder architecture is used. The encoder processes input patches with masked regions, while the decoder reconstructs the missing areas. Once the training is complete, the encoder serves as the feature extractor, generating feature vectors that capture contextual information from the surrounding texture.

In rotation prediction, two identical CNNs independently process input patches and their rotated counterparts, producing feature vectors. The process consists of two key steps: feature extraction and rotation prediction. After extracting the features, the vectors are concatenated and passed through a fully connected dense layer, which acts as the prediction module. This module calculates the relative rotation between the two input patches and outputs a single rotation prediction. The feature extraction component of this architecture is then integrated into the K-views algorithm for texture classification.

For the connection prediction task, the proposed approach uses Siamese networks. Two patches are processed through identical CNNs to produce their respective feature vectors. These vectors are concatenated and passed to a classifier that determines whether the patches are spatially connected. The encoder from this Siamese network is subsequently used to extract spatially aware feature representations, which are utilized within the K-views algorithm to enhance its ability to capture structural relationships in texture patterns.

4.2. Integrating feature extractors with the K-views algorithm

The integration of feature extractors with the K-views algorithm aims to enhance its ability to classify textures by leveraging robust, learned representations. Various feature extraction methods, as explained in Section 2 (including GLCM, LBP, PCA, SIFT, CNN autoencoder, variational autoencoder, and Siamese networks) and Section 4 (including connection prediction, image inpainting, and rotation prediction), are used to extract features from patches. These features are subsequently used in the clustering and classification stages of the K-views algorithm.

The clustering process applies Euclidean distance to group similar features, and this measure is consistent across all feature extraction methods. Below is the step-by-step process, including details on the clustering approach and how the distance measure is applied (Algorithm 2):

Algorithm 2. The K-views algorithm with patch-wise feature extraction.

```
Input: Image $I$, trained encoder, number of characteristic views $K$, view size $Vs$, number of clusters (classes) $N$
Output: Classified texture class.
Steps:
1. Divide the input image $I$ into views (patches) of size $Vs$.
2. For each patch in $I$:
   - Use the feature extraction methods explained in Section 2 and Section 4 to extract feature
representations for the patch.
3. Aggregate all extracted feature representations into a feature set.
4. Apply the clustering algorithm on the feature set to identify $N$ clusters in the feature space.
   - Euclidean distance is used as the distance measure for clustering the feature vectors.
5. Assign each patch to one of the $N$ clusters based on its extracted feature representation.
6. Determine the texture class for the image:
   - Identify the cluster with the highest number by counting assigned patches.
   - Assign the texture class corresponding to this cluster with the highest count as the final classification for the image.
Return: Classified texture class.
```

5. Experimental results

The evaluation of the K-views algorithm was performed on three benchmark texture datasets: KTH-Tips, Kylberg, and UIUC, integrating multiple feature extraction techniques for comparison. Alongside traditional and deep learning methods, self-supervised learning techniques—image inpainting, rotation prediction, and the proposed connection prediction method—were analyzed, showcasing their capability to enhance texture classification without requiring labeled data. A summary of the datasets including the key statistics is shown in Table 1.

Table 1. Summary of the datasets.

Dataset	Number of classes	Total number of images	Number of samples for each class	Image size
KTH-TIPS	10	810	81	200 x 200
Kylberg	28	4480	160	576 x 576
UIUC	25	1000	40	331 x 331

5.1. KTH-TIPS dataset

The KTH-TIPS dataset extends the CUReT database by introducing variations in scale, pose, and illumination for 10 material classes, resulting in 810 images with 81 samples per class and a patch size of 200 × 200 pixels (Figure 6). Images were captured at nine different scales and under three distinct poses and illumination conditions using an Olympus digital camera. While some images contain artifacts such as poor focus, the dataset's diversity makes it a valuable resource for assessing the performance of texture classification algorithms in real-world scenarios.

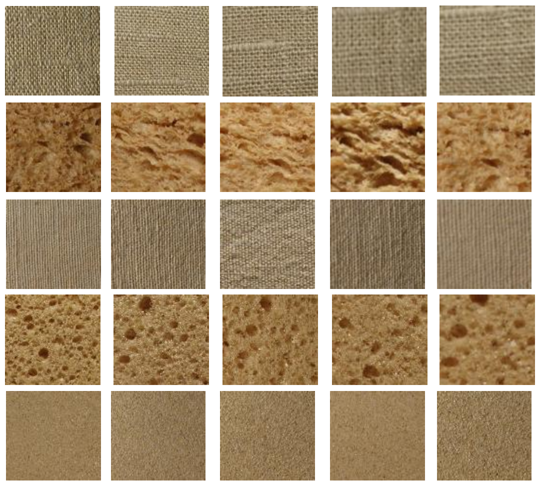

Figure 6. Examples of linen, brown bread, cotton, a sponge, and sandpaper, respectively, retrieved from the KTH-TIPS dataset [13].

On the KTH-Tips dataset, integrating feature extraction methods into the K-views algorithm demonstrated clear performance differences, as shown in Table 2. Among the self-supervised approaches, rotation prediction achieved the highest performance, with an accuracy of 72%, precision of 0.67, recall of 0.64, and an F1-score of 0.65. The connection prediction method followed closely with 71% accuracy and an F1-score of 0.63. The supervised Siamese networks achieved perfect classification with 100% accuracy, precision, recall, and F1-score. Other methods like the CNN-based feature extractor also performed strongly, achieving an F1-score of 0.77, and well-balanced precision and recall values of 0.77 and 0.76, respectively.

Table 2. Experimental results for the KTH-TIPS dataset.

Model	Accuracy	Precision	Recall	F1
K-views algorithm	0.54	0.44	0.40	0.42
+ GLCM	0.54	0.45	0.40	0.43
+ LBP	0.59	0.51	0.47	0.49
+ PCA	0.57	0.48	0.44	0.46
+ SIFT	0.57	0.49	0.45	0.47
+ CNN autoencoder	0.63	0.56	0.52	0.54
+ Variational autoencoder	0.65	0.58	0.54	0.56
+ Image inpainting	0.70	0.64	0.61	0.63
+ Rotation prediction	0.72	0.67	0.64	0.65
+ Connection prediction	0.71	0.65	0.62	0.63
+ CNN-based feature extractor	0.81	0.77	0.76	0.77
+ Siamese networks	1.00	1.00	1.00	1.00

5.2. Kylberg dataset

The Kylberg texture dataset is a benchmark designed to test texture classification algorithms under varying conditions. It comprises 4480 images representing 28 texture classes, with 160 samples per class and a patch size of 576 × 576 pixels (Figure 7). The dataset includes textures from materials such as fabric, stone, and wood, captured at different scales, angles, and lighting conditions. These variations create a challenging environment for assessing algorithm performance, particularly in scenarios with changes in illumination, scale, and viewpoint.

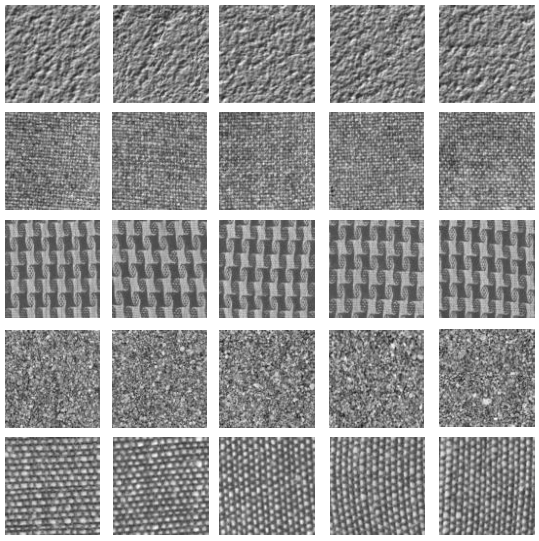

Figure 7. Examples of a ceiling, blanket, scarf, sand, and cushion, respectively, retrieved from the Kylberg dataset [14].

For the Kylberg dataset, rotation prediction proved to be the most effective self-supervised method, as reported in Table 3, achieving an accuracy of 75%, a precision of 0.71, a recall of 0.68, and an F1-score of 0.69, followed by image inpainting with an F1-score of 0.60 and the proposed connection prediction method with an F1-score of 0.59. Traditional approaches like GLCM and LBP performed moderately well, with F1-scores of 0.64 and 0.58, respectively. Siamese networks achieved near-perfect results, with 95% accuracy, 0.94 precision and recall, and an F1-score of 0.94.

Table 3. Experimental results for the Kylberg dataset.

Model	Accuracy	Precision	Recall	F1
K-views algorithm	0.55	0.46	0.42	0.44
+ GLCM	0.71	0.65	0.62	0.64
+ LBP	0.67	0.60	0.57	0.58
+ PCA	0.57	0.48	0.44	0.46
+ SIFT	0.63	0.56	0.52	0.54
+ CNN autoencoder	0.58	0.50	0.46	0.48
+ Variational autoencoder	0.66	0.59	0.56	0.58
+ Image inpainting	0.68	0.62	0.58	0.60
+ Rotation prediction	0.75	0.71	0.68	0.69
+ Connection prediction	0.67	0.61	0.57	0.59
+ CNN-based feature extractor	0.85	0.82	0.80	0.81
+ Siamese networks	0.95	0.94	0.94	0.94

5.3. UIUC dataset

The UIUC texture dataset, developed at the University of Illinois Urbana-Champaign, contains 1000 images across 25 texture classes, with 40 samples per class and a patch size of 331 × 331 pixels (Figure 8). The dataset features textures from a variety of materials, including metal, fabric, and grass, captured under diverse lighting conditions. To ensure consistent scale, all images are aligned and resized, facilitating straightforward comparisons of algorithmic performance. The UIUC dataset is frequently employed in texture classification tasks, serving as a standard benchmark for evaluating and comparing methods.

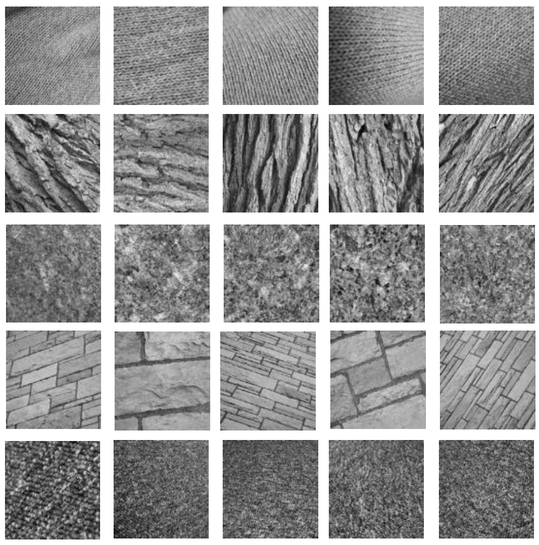

Figure 8. Examples of knit, bark, granite, wall, and carpet textures, respectively, retrieved from the UIUC dataset [15].

On the UIUC dataset, the Siamese network achieved the best overall performance, significantly outperforming all other methods with 74% accuracy, and particularly strong precision of 0.68 and recall of 0.66, leading to the highest F1-score of 0.67, as presented in Table 4. Among self-supervised methods, rotation prediction stood out with a notable accuracy of 61%, and a balanced trade-off between precision and recall (0.53 and 0.49, respectively), resulting in an F1-score of 0.51. While connection prediction performed slightly worse with 56% accuracy and an F1-score of 0.45, it still outpaced simpler enhancements like GLCM and LBP. Other methods like CNN-based feature extraction demonstrated competitive results, achieving an F1-score of 0.56.

Table 4. Experimental results for the UIUC dataset.

Model	Accuracy	Precision	Recall	F1
K-views algorithm	0.54	0.45	0.40	0.43
+ GLCM	0.53	0.44	0.39	0.41
+ LBP	0.55	0.46	0.42	0.44
+ PCA	0.54	0.45	0.41	0.43
+ SIFT	0.55	0.46	0.41	0.44
+ CNN autoencoder	0.58	0.49	0.45	0.47
+ Variational autoencoder	0.58	0.50	0.46	0.48
+ Image inpainting	0.55	0.46	0.42	0.44
+ Rotation prediction	0.61	0.53	0.49	0.51
+ Connection prediction	0.56	0.48	0.43	0.45
+ CNN-based feature extractor	0.64	0.57	0.54	0.56
+ Siamese networks	0.74	0.68	0.66	0.67

The results highlight the effectiveness of self-supervised learning methods in enhancing the K-views algorithm. While rotation prediction consistently outperformed other self-supervised methods, the proposed connection prediction technique also exhibited strong performance, particularly on the KTH-Tips and the Kylberg datasets.

Traditional methods like GLCM and LBP, although foundational, were surpassed by deep learning-based and self-supervised methods. Supervised approaches, particularly Siamese networks, demonstrated unmatched performance but required labeled data, highlighting the trade-off between supervision and scalability.

6. Conclusions

This study presents a comprehensive exploration of feature extraction techniques integrated into the basic K-views algorithm for texture analysis. Our methodology incorporated both traditional and advanced feature extraction approaches, evaluated across three benchmark datasets: KTH-TIPS, Kylberg, and UIUC. The results demonstrate significant improvements in texture classification performance, validating the effectiveness of our enhancements.

Traditional methods like GLCM and LBP provided foundational insights into texture properties but were limited in their ability to capture complex, high-dimensional texture relationships. Dimensionality reduction techniques, such as PCA, offered computational efficiency but showed restricted discriminative power. The SIFT contributed robustness against image transformations but achieved only moderate improvements in classification metrics.

Deep learning-based approaches significantly advanced the performance of the K-views algorithm. Autoencoder-based methods, including CNNs and VAEs, captured high-level texture representations, surpassing the capabilities of traditional methods. Self-supervised approaches, such as image inpainting, rotation prediction, and the proposed connection prediction, achieved notable improvements without requiring labeled data, highlighting their potential for scalable applications. Among these, the rotation prediction consistently outperformed other self-supervised methods, while the novel connection prediction technique provided a unique perspective by leveraging inter-patch relationships to enrich the feature extraction process.

The integration of CNN-based feature extractors and Siamese networks had a significant impact on the classification performance. CNN-based feature extraction achieved consistently high performance, demonstrating the versatility of deep learning in capturing intricate texture patterns. Siamese networks emerged as the most effective feature extraction method, achieving perfect clustering performance on the KTH-TIPS dataset and substantial improvements on the Kylberg and UIUC datasets. Their ability to discern similarities and differences between texture patches enabled the creation of highly separable feature spaces, resulting in superior classification results.

Self-supervised methods demonstrated significant performance gains, particularly given their independence from labeled data. Our findings suggest that combining the strengths of these approaches offers an optimal strategy for texture analysis. Specifically, we suggest that training with self-supervised methods to extract robust feature representations, followed by fine-tuning using the K-views algorithm with Siamese networks on a small labeled dataset, might give the optimum results. This two-stage approach balances scalability with high performance, reducing the reliance on extensive labeled data while leveraging advanced deep learning techniques for fine-grained classification.

Future work could explore implementing and validating this two-stage approach on larger and more diverse datasets, using a small, labeled dataset, as well as assessing its applicability in real-world texture classification tasks.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

Min Wang and Chih-Cheng Hung are the editorial board members for Applied Computing and Intelligence and were not involved in the editorial review and the decision to publish this article.

References

1. A. Humeau-Heurtier, Texture feature extraction methods: a survey, *IEEE Access*, **7** (2019), 8975–9000. https://doi.org/10.1109/ACCESS.2018.2890743
2. A. Humeau-Heurtier, Color texture analysis: a survey, *IEEE Access*, **10** (2022), 107993–108003. https://doi.org/10.1109/ACCESS.2022.3213439
3. L. Armi, S. Fekri-Ershad, Texture image classification based on improved local quinary patterns, *Multimed. Tools Appl.*, **78** (2019), 18995–19018. https://doi.org/10.1007/s11042-019-7207-2
4. J. Shotton, J. Winn, C. Rother, A. Criminisi, Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context, *Int. J. Comput. Vis.*, **81** (2009), 2–23. https://doi.org/10.1007/s11263-007-0109-1
5. C. C. Hung, E. Song, Y. Lan, *Image texture analysis: foundations, models and algorithms*, Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-13773-1
6. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. Efros, Context encoders: feature learning by inpainting, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 2536–2544. https://doi.org/10.1109/CVPR.2016.278

7. S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, arXiv: 1803.07728. https://doi.org/10.48550/arXiv.1803.07728
8. R. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, *IEEE Trans. Syst. Man Cy.*, **SMC-3** (1973), 610–621. https://doi.org/10.1109/TSMC.1973.4309314
9. T. Ojala, M. Pietikäinen, D. Harwood, A comparative study of texture measures with classification based on featured distributions, *Pattern Recogn.*, **29** (1996), 51–59. https://doi.org/10.1016/0031-3203(95)00067-4
10. D. Lowe, Object recognition from local scale-invariant features, *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, 1150–1157. https://doi.org/10.1109/ICCV.1999.790410
11. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *P. IEEE*, **86** (1998), 2278–2324. https://doi.org/10.1109/5.726791
12. J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "Siamese" time delay neural network, *Proceedings of the 7th International Conference on Neural Information Processing Systems*, 1993, 737–744.
13. *M. Fritz, E. Hayman, B. Caputo, J. Eklundh, The KTH-TIPS database,* KTH, 2004. Available from: https://www.csc.kth.se/cvap/databases/kth-tips/kth_tips.pdf.
14. G. Kylberg, *Kylberg texture dataset v. 1.0*, Uppsala: Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, 2011.
15. S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, *IEEE Trans. Pattern Anal.*, **27** (2005), 1265–1278. https://doi.org/10.1109/TPAMI.2005.151
16. C. Zhang, Z. Zhang, A survey of recent advances in face detection, Technical report: MSR-TR-2010-66.
17. B. Manjunath, W. Ma, Texture features for browsing and retrieval of image data, *IEEE Trans. Pattern Anal.*, **18** (1996), 837–842. https://doi.org/10.1109/34.531803
18. T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, *IEEE Trans. Pattern Anal.*, **24** (2002), 971–987. https://doi.org/10.1109/TPAMI.2002.1017623
19. R. Haralick, L. Shapiro, Image segmentation techniques, *Computer Vision, Graphics, and Image Processing*, **29** (1985), 100–132. https://doi.org/10.1016/S0734-189X(85)90153-7
20. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature*, **521** (2015), 436–444. https://doi.org/10.1038/nature14539
21. X. Dong, H. Zhou, J. Dong, Texture classification using pair-wise difference pooling-based bilinear convolutional neural networks, *IEEE Trans. Image Process.*, **29** (2020), 8776–8790. https://doi.org/10.1109/TIP.2020.3019185
22. C. Doersch, A. Gupta, A. Efros, Unsupervised visual representation learning by context prediction, *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, 1422–1430. https://doi.org/10.1109/ICCV.2015.167
23. S. Fati, E. Senan, A. Azar, Hybrid and deep learning approach for early diagnosis of lower gastrointestinal diseases, *Sensors*, **22** (2022), 4079. https://doi.org/10.3390/s22114079
24. S. Nair, M. Subaji, A novel feature fusion for the classification of histopathological carcinoma images, *Int. J. Adv. Comput. Sci.*, **4** (2023), 688–697. https://doi.org/10.14569/ijacsa.2023.0140972
25. P. Sethy, N. Barpanda, A. Rath, S. Behera, Deep feature based rice leaf disease identification using support vector machine, *Comput. Electron. Agr.*, **175** (2020), 105527. https://doi.org/10.1016/j.compag.2020.105527

26. H. Abdi, L. Williams, Principal component analysis, *WIREs Comput. Stat.*, **2** (2010), 433–459. https://doi.org/10.1002/wics.101
27. J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, In: *Artificial neural networks and machine learning–ICANN 2011*, Berlin: Springer, 2011, 52–59. https://doi.org/10.1007/978-3-642-21735-7_7
28. D. Kingma, M. Welling, Auto-encoding variational bayes, arXiv: 1312.6114. https://doi.org/10.48550/arXiv.1312.6114
29. *R. Khandelwal, One-shot learning with Siamese network: an intuitive explanation of Siamese network*, The Startup, 2021. Available from: https://medium.com/swlh/one-shot-learning-with-siamese-network-1c7404c35fda.