





Article

A Comparative Survey of Vision Transformers for Feature Extraction in Texture Analysis

Leonardo Scabini ^{1,*}, Andre Sacilotti ², Kallil M. Zielinski ¹, Lucas C. Ribas ³, Bernard De Baets ⁴
and Odemir M. Bruno ^{1,2}

¹ São Carlos Institute of Physics, University of São Paulo, São Carlos 13560-970, SP, Brazil; bruno@ifsc.usp.br (O.M.B.)

² Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos 13566-590, SP, Brazil; andre.sacilotti@usp.br

³ Institute of Biosciences, Humanities and Exact Sciences, São Paulo State University, São José do Rio Preto 15054-000, SP, Brazil; lucas.ribas@unesp.br

⁴ KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, 9000 Ghent, Belgium; bernard.debaets@ugent.be

* Correspondence: scabini@ifsc.usp.br

Abstract

Texture, a significant visual attribute in images, plays an important role in many pattern recognition tasks. While Convolutional Neural Networks (CNNs) have been among the most effective methods for texture analysis, alternative architectures such as Vision Transformers (ViTs) have recently demonstrated superior performance on a range of visual recognition problems. However, the suitability of ViTs for texture recognition remains underexplored. In this work, we investigate the capabilities and limitations of ViTs for texture recognition by analyzing 25 different ViT variants as feature extractors and comparing them to CNN-based and hand-engineered approaches. Our evaluation encompasses both accuracy and efficiency, aiming to assess the trade-offs involved in applying ViTs to texture analysis. Our results indicate that ViTs generally outperform CNN-based and hand-engineered models, particularly when using strong pre-training and in-the-wild texture datasets. Notably, BeiTv2-B/16 achieves the highest average accuracy (85.7%), followed by ViT-B/16-DINO (84.1%) and Swin-B (80.8%), outperforming the ResNet50 baseline (75.5%) and the hand-engineered baseline (73.4%). As a lightweight alternative, EfficientFormer-L3 attains a competitive average accuracy of 78.9%. In terms of efficiency, although ViT-B and BeiT(v2) have a higher number of GFLOPs and parameters, they achieve significantly faster feature extraction on GPUs compared to ResNet50. These findings highlight the potential of ViTs as a powerful tool for texture analysis while also pointing to areas for future exploration, such as efficiency improvements and domain-specific adaptations.

Keywords: texture analysis; vision transformers; transfer learning; computer vision; deep learning



Academic Editors: Raimondo Schettini and Guanghui (Richard) Wang

Received: 23 June 2025

Revised: 20 August 2025

Accepted: 26 August 2025

Published: 5 September 2025

Citation: Scabini, L.; Sacilotti, A.; Zielinski, K.M.; Ribas, L.C.; De Baets, B.; Bruno, O.M. A Comparative Survey of Vision Transformers for Feature Extraction in Texture Analysis. *J. Imaging* **2025**, *11*, 304. <https://doi.org/10.3390/jimaging11090304>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Computer vision (CV) has become an extensive subfield of Artificial Intelligence, especially after the proliferation of deep learning over the past decade. One of the subfields of CV, which dates back to the 1960s, is texture analysis. For digital images, one abstract definition is that texture elements emerge from the local intensity constancy and/or variations of pixels producing spatial patterns roughly independently at different scales [1].

Although general vision involves the combination of several other aspects such as shape and depth, texture alone is a fundamental characteristic that can suffice to solve many problems. Therefore, over the past decades several texture analysis methods have been proposed [2,3]. These works have led to many applications in industrial inspection [4] and medical imaging [5], to name but a few. Figure 1 illustrates the usual texture analysis approach: a model extracts relevant information from a texture image to compose an image representation, which is used for pattern recognition tasks that rely on these textures.

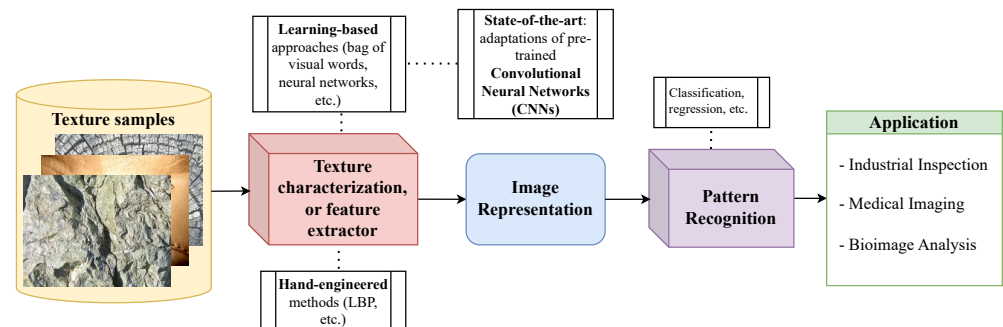


Figure 1. The usual pipeline in texture analysis. Texture samples are first processed by a feature extraction stage (texture characterization), which can use either hand-engineered methods or learning-based techniques. These techniques produce intermediate image representations, which are then used in pattern recognition tasks such as classification or regression. This pipeline can be employed in a variety of domains, including industrial inspection, medical imaging, and bioimage analysis.

Deep learning-based models for general vision tasks have been advancing rapidly. This revolution in the field started with CNNs [6–8], a powerful neural architecture that still dominates many CV areas. However, CNNs may fail to achieve state-of-the-art (SOTA) performance in texture recognition tasks in comparison to hand-engineered approaches [1,9]. More recently, ViTs have started to dominate the CV literature while challenging CNNs, especially on image classification tasks [10,11]. Nevertheless, little is known regarding the applicability of ViTs to texture analysis. This is a significant research gap, particularly because ViTs do not possess the spatial inductive biases (like translation equivariance and locality) that are fundamental to CNNs. While these biases are considered advantageous for many vision tasks, their absence in ViTs raises critical questions about how these models learn to represent and recognize textural patterns. Furthermore, texture datasets are often limited in size, which represents a considerable challenge for data-hungry models such as ViTs.

In order to overcome this gap, this paper presents a comprehensive and large-scale study to assess the capabilities of Vision Transformers for texture analysis. Our central objective is to systematically evaluate the potential of using pre-trained ViTs as powerful feature extractors for a wide array of texture recognition tasks. We aim to understand how these “foundation models”, which have proven so effective for general vision, perform when applied to the specific domain of texture and how they compare against established CNN and hand-engineered baselines.

To achieve this, we conduct an extensive analysis of twenty-one distinct ViT variants. These models are carefully selected to cover a wide range of architectural designs and pre-training strategies, including both supervised and self-supervised paradigms, on large-scale datasets like ImageNet-1k and ImageNet-21k. Our methodology employs transfer learning: We use the pre-trained weights of these foundation models without fine-tuning, utilizing them as frozen feature extractors. A linear classifier is then trained on top of these extracted features for the final texture recognition task.

The scope of our evaluation is designed to encompass a variety of challenging scenarios, like tasks that measure robustness to changes in rotation, scale, and illumination. In addition, we evaluate the ability to discriminate between different and complex texture categories, including color textures, material textures, and descriptive texture attributes found in the wild. Through this investigation, we provide a clear benchmark and an in-depth analysis of the strengths and weaknesses of ViTs for texture analysis, offering valuable insights for future research in the field.

2. Background

2.1. Vision Transformers

The transformer architecture [12] is an effective deep learning mechanism for machine translation tasks with a more parallelizable architecture because it processes all input tokens simultaneously using self-attention, unlike recurrent architectures that rely on sequential token processing. The first architecture was designed as a stack of encoders and decoders, containing two main structures: multi-head self-attention (MSA) and a Feed Forward Network. First of all, consider a set of tokens and their embeddings (e.g., words and word embeddings) combined into a matrix X . The first step is to transform these inputs by projecting them using linear layers, obtaining a query matrix $Q_i = XW_{Q_i}$, where W_{Q_i} represents the query weights and matrices $K_i = XW_{K_i}$ and $V_i = XW_{V_i}$ represent the keys and values, respectively, and their corresponding weights. Here, the index i refers to the i -th attention head in the MSA mechanism, which has its own learnable projection matrices $W_{Q_i}, W_{K_i}, W_{V_i}$. The self-attention mechanism for all tokens is given by

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d}}\right) V_i, \quad (1)$$

where the softmax function is taken over the horizontal axis and d is the hidden dimension of the model (embedding size). A single self-attention mechanism is referred to as an attention head, and MSA is achieved by stacking s attention heads in parallel, each with individual trainable weights:

$$\text{MSA}(Q, K, V) = [\text{Attention}(Q_1, K_1, V_1); \dots; \text{Attention}(Q_s, K_s, V_s)] W_O, \quad (2)$$

where $[\cdot]$ denotes the concatenation of the self-attentions and W_O represents the weights for a final linear projection after the concatenation. A Feed Forward Network is applied over the output of the MSA, which is a simple MLP with two layers. Additionally, layer normalization and residual connections are added between the layers, finally composing a transformer block. A standard transformer network is then the combination of a series of transformer blocks, followed by an output layer depending on the task at hand.

More recently, ViTs [10,11] have been dominating the CV literature, challenging CNNs. They have been applied to a lot of different visual tasks, including, but not limited to, image classification [10,11,13,14], object detection [15,16], image segmentation [17], and super-resolution [18]. Results demonstrate that ViTs achieve SOTA performance on CV tasks, on par with CNNs. Figure 2a shows the general structure of a ViT and some of the main architectural choices in recent works.

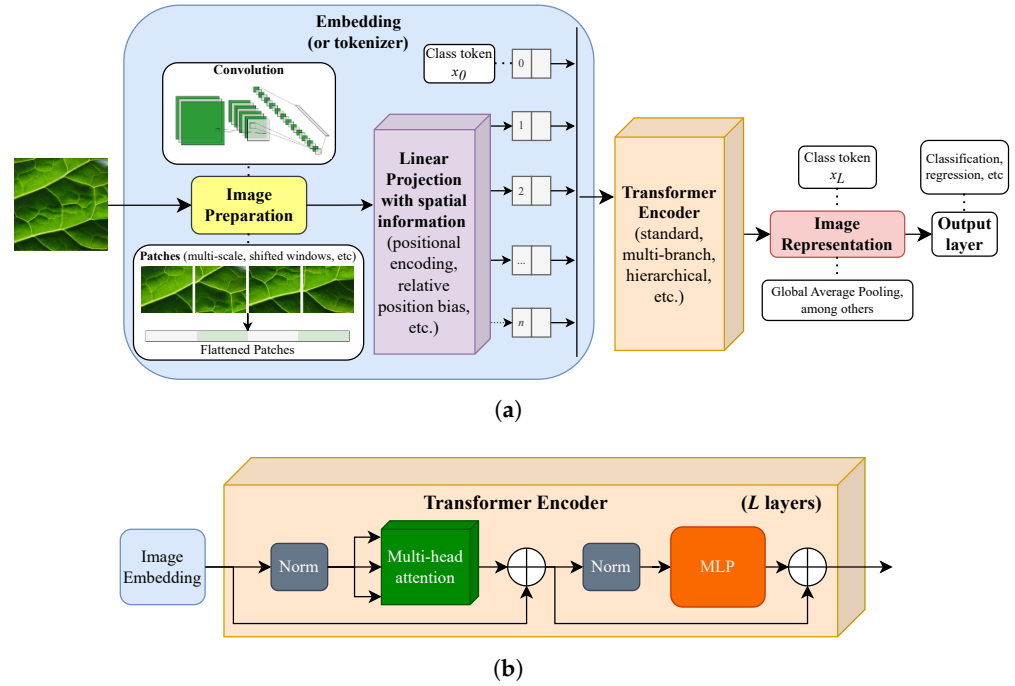


Figure 2. The general elements of a Vision Transformer (a). One of its most important modules is the image embedding (also known as the tokenizer), which is responsible for preparing the pixels in a way that transformer encoders (b) can learn and extract meaningful visual patterns. (a) General structure of a Vision Transformer and some of the different options that can be selected at each stage. (b) The most common type of transformer encoder (adapted from ref. [10]).

ViTs offer an efficient way to adapt the transformer architecture to images by representing them as a sequence of 2D patches. Consider an RGB image $I \in \mathbb{R}^{w \times h \times 3}$, with width w and height h , fed into a ViT backbone (as in ref. [10]) $B = (T_1, \dots, T_L)$, consisting of L sequential transformer blocks. The image I is firstly reshaped into a sequence of flattened 2D patches $I_p \in \mathbb{R}^{n \times (3p^2)}$, where 3 represents the RGB colorspace, $p \times p$ is the patch size, and $n = \frac{wh}{p^2}$ represents the number of patches (input sequence length). I_p is then projected using a trainable linear projection (linear layer) $E \in \mathbb{R}^{(3p^2) \times d}$, where d is the constant hidden size of the transformer architecture. The patch embeddings I_e are obtained by adding (element-wise sum) a positional encoding layer $E_{pos} \in \mathbb{R}^{(n+1) \times d}$, including spatial inductive bias, into the transformer. I_e is then fed into a sequence of L transformer encoders (as shown in Figure 2b), which can be trained by adding an MLP classifier at the end. This is possible since the ViT includes a learnable class embedding (or class token),

$$x_l^0 \in \mathbb{R}^d, \quad (3)$$

among the embeddings of each layer l , which encodes the information in a one-dimensional vector with d features. Therefore, the class embedding at the last transformer layer, x_L^0 , serves as an image representation where a classification head can be attached. In this work, we focus on this embedding as an off-the-shelf image representation for texture recognition tasks using simpler/linear classifiers, so the common MLP head of ViTs is not considered.

2.2. Texture Analysis

Texture analysis is a subfield of CV with roots in the 1960s. Although an abstract concept, with no widely accepted formal definition, texture refers to the perceived surface properties or structure of objects, which may include roughness, smoothness, coarseness, or fineness. It can also be seen as a pattern of local variations in color and brightness. The

human visual system is adept at recognizing and distinguishing textures, allowing us to differentiate between many things in our environment. Therefore, recognizing textures in digital images is critical to solving many CV problems and tasks such as feature extraction, classification, segmentation, and synthesis, among others. As a result, a variety of methods have been developed over the past decades [2,3], paving the way for potential applications in fields such as industrial inspection [4] and medical imaging [5], among others.

For many years, the predominant approach to texture recognition was based on hand-engineered models or features to describe textures. Mathematical methods for the description of textural patterns usually consider properties such as statistics [19], frequency [20], complexity/fractality [1,21], and others [22]. Statistical methods investigate local measures based on grayscale co-occurrences, the most widely used being local binary patterns (LBPs) [19], which have influenced various subsequent techniques. Another approach to texture analysis involves transforming the image into the frequency domain, where various methods such as Gabor filters [20] have been proposed. Complexity approaches fall within the model-based paradigm, such as methods based on fractal dimension [21] or network science [1,23].

After the popularization of learning-based models for object recognition, many such methods were also specifically designed for tackling texture. For instance, there have been various studies involving deep CNNs for texture recognition by using transfer learning. The most common approach for the transfer learning of foundation vision models is to fine-tune a pre-trained network for the desired task. However, even if these models are pre-trained, it is impossible to predict how much fine-tuning data would be necessary to achieve satisfactory performance. In the case of texture analysis, previous works have shown [9] that fine-tuning or training these models from scratch may result in poor performance in texture recognition, particularly due to overfitting caused by a lack of large-scale texture datasets.

Some studies explore the transfer learning of deep CNNs by using convolutional layers only for extraction of texture features, freezing their parameters, and using a dedicated classifier trained separately. This approach is also known as “features-off-the-shelf”, or deep convolutional activation features [24], and is a simple and fast way to transfer learning from foundation vision models. Cimpoi et al. [25] proposed one of the first contributions on the subject, comparing the efficiency of different deep CNN architectures and approaches for feature extraction. Subsequently, many works have been proposed following these principles. One of the latest techniques, named Random encoding of Aggregated Deep Activation Maps (RADAM) [26], performs multi-depth deep feature aggregation and trains randomized auto-encoders for each image to produce an encoded representation. This method does not fine-tune the CNN backbone, and results demonstrate that these locally learned representations provide SOTA performance in texture recognition.

Another approach consists of end-to-end architectures that enable the training of new texture-specific modules/layers along with the fine-tuning of pre-trained CNN backbones. Zhang et al. [27] proposed an orderless encoding layer on top of a deep CNN, called Deep Texture Encoding Network (Deep-TEN), which allows for images of arbitrary size. Yang et al. [28] proposed DFAEN (Double-order Knowledge Fusion and Attentional Encoding Network), which aggregates first- and second-order information for encoding texture features. Fine-tuning is needed in these methods to adapt the backbone to the new architecture along with a new classification layer since they contain new randomly initialized parameters.

As discussed above, SOTA texture recognition models [26,28] adopt pre-trained CNN backbones for texture feature extraction, aggregation, and encoding, achieving promising results. However, the fixed/limited size of a CNN’s receptive field may struggle to model the correlations among global features and long-distance pixel relationships,

which are critical for many tasks involving textures. On the other hand, the transformer architecture excels at capturing these patterns, which suggests they could be valuable alternatives [10,29,30]. Nevertheless, since ViT architectures are quite recent, only a few studies have specifically focused on texture aspects. For instance, some works explore textures in transformers for image super-resolution [31,32] and remote sensing [33]. For texture recognition, our focus in this work, a couple of works have analyzed ViTs in specific cases. In ref. [34], the authors used ViTs for the recognition of steel texture blocks, showing that a custom transformer architecture can obtain a higher accuracy than a standard CNN and machine learning models. Another study analyzed the ViT architecture in the field of building and construction material recognition [35], demonstrating the capability to deal with imbalanced datasets, achieving a higher accuracy compared to classical CNNs. A more recent work [36] introduces a hybrid transformer model for the localization of anomalies on industrial textured surfaces.

Despite some efforts to adopt ViT models for texture analysis, there is a lack of understanding of how or why this architecture works for different types of textures and the impacts and differences of the several ViT variants quickly emerging in the literature. No study has analyzed ViTs for texture recognition in general, considering well-known benchmarks, robustness issues, the impacts of different architectural choices, pre-trainings, and so on. Prior research has focused on specific problems related to texture analysis, which is not sufficient to promote ViTs as the next SOTA in this area. Therefore, we focus on these aspects by proposing a comprehensive evaluation of a variety of known ViTs when applied to a wide range of texture recognition tasks.

2.3. Selected ViTs

The literature on ViTs has been quickly advancing in the past three years due to the success of one of their first implementations for image recognition [10]. It would be impossible to cover here all the models proposed under the ViT umbrella in this period. For this purpose, we select a set of different ViT variants, considering the most prominent differences in their architecture, pre-training, and computational budget. Table 1 shows the main properties of the selected variants. Additionally, we give a more detailed description of each ViT below.

- **ViT-B/16** [10]: This was one of the first successful computer vision variants of the transformer model, proposed by Dosovitskiy et al. The model employed in this work, ViT-B/16, has a base size “B” encompassing 12 layers, a hidden dimension of size 768, and 12 attention heads. This configuration processes input images by dividing them into non-overlapping patches of 16×16 pixels, hence the B/16 designation. Each patch is then linearly embedded into a flat vector and passed through the transformer layers for further processing.
- **CoaT** [37]: Co-scale conv-attentional image Transformers (CoaTs) contain two mechanisms to improve ViTs in image classification: (i) The co-scale mechanism, maintaining separate encoder branches at different scales while allowing for attention across these scales. A serial and a parallel block were created to perform fine-to-coarse, coarse-to-fine, and cross-scale image modeling. (ii) A conv-attention module that incorporates convolutions in the factorized attention module for relative position embeddings, resulting in a considerably improved computational cost compared to traditional self-attention layers in transformers. The authors introduced two architectures: CoaT-Lite, which exclusively uses serial blocks to sequentially process down-sampled image features, and CoaT, which incorporates both serial and parallel blocks with the co-scale mechanism. Additionally, the authors test CoaT and CoaT-Lite across various model

sizes: Tiny, Mini, Small, and Medium. Consequently, CoaT-Li-Mi and CoaT-Mi refer to the CoaT-Lite Mini and CoaT Mini variants, respectively.

- **MobileViT-S** [38]: The design of this architecture targets mobile vision applications, focusing on compactness, general purpose, and minimal latency. For this purpose, the network integrates key characteristics from CNNs, such as spatial inductive biases and reduced sensitivity to data augmentation, with those from ViTs, including input-adaptive weighting and global processing capabilities [38]. By incorporating properties from both CNNs and ViTs, MobileViT achieves a discriminative representation using a low number of parameters and simple training approaches, such as basic augmentation techniques. The MobileViT model has three size variations typically used in mobile applications: small, extra small, and extra extra small. In this work, we used the small version, which has 5.6 million parameters.
- **MobileViTv2** [39]: Although MobileViT models exhibit high performance and have few parameters compared to light-weight CNNs, they still face the issue of high latency, primarily due to the multi-headed self-attention. To overcome this limitation, MobileViTv2, an enhanced version of MobileViT, introduces a separable self-attention mechanism with linear complexity that calculates context scores relative to a latent token.
- **EfficientFormer** [40]: The EfficientFormer, a family of models, introduces a new dimension-consistent design paradigm for Vision Transformers, incorporating a simple but efficient latency-driven slimming technique [40]. Instead of reducing the number of parameters or computations, EfficientFormer networks are designed to optimize inference speed. Within this family, the EfficientFormer-L1 is the fastest model, whereas the EfficientFormer-L3 and EfficientFormer-L7 are the largest models, offering better performances.
- **ViT-B/16-DINO** [41]: Distillation with NO labels (DINO) is a self-supervised learning approach for training vision models, such as ViTs, without the need for labeled data. It relies on a teacher–student framework with a distillation loss and noisy labels to guide the student model. In this work, we consider the ViT-B/16-DINO variant, which corresponds to the DINO approach applied to the ViT-B/16 model using IN-1k.
- **ViT-B/16-SAM** [42]: The incorporation of the Sharpness-Aware Minimizer (SAM) into the ViT-B/16 model is an approach that explicitly smooths the loss geometry during training, leading to improved generalization capabilities. By utilizing SAM, the enhanced ViT-B/16 model not only achieved better accuracy and robustness compared to ResNets with similar and larger sizes, but also demonstrated effective training with (momentum) SGD.
- **DeiT-B/16** [43]: Standing for Data-efficient image Transformers (DeiT), this method improves the data efficiency of ViTs by employing knowledge distillation, a technique that transfers knowledge from a larger pre-trained teacher model to a smaller student model. DeiT-B/16 is a specific configuration of the DeiT model that adopts the same 16×16 patch size, 12 layers, and 768 hidden dimension size as its counterpart in the ViT family.
- **DeiT3-B/16** [11]: The DeiT3 method introduces a new training procedure for ViT architectures and is an upgrade of the previous DeiT. Key experiments conducted by the author involve the following: adopting a binary cross-entropy loss for IN-1k training; comparing simple random cropping to random resize cropping when pre-training on larger datasets such as IN-21k; and training models at lower resolutions to reduce the train–test discrepancy. DeiT3-B/16 refers to a ViT-B/16 model trained using the DeiT3 methodology. Here, we employ two variants of this model: with IN-1k pre-training or using IN-21k then fine-tuning on IN-1k.

- **CrossViT-B [44]:** The Cross-Attention Multi-Scale Vision Transformer (CrossViT) introduces a dual-branch transformer architecture. In one branch, the model processes fine-grained small patches from the image, while the other branch focuses on coarse-grained large patches. This design aims to generate more significant features by incorporating information from different scales. Also, the work proposes a token fusion mechanism with linear complexity, which combines the class token from one branch with the other patches and vice versa. The architecture is trained based on the approach outlined in DeiT [11].
- **ConViT [45]:** The ConViT architecture tries to mimic the convolutional inductive bias, introducing a new attention scheme, Gated Positional Self-Attention (GPSA). This mechanism forces the attention to initialize following an almost convolutional configuration, adding parameters related to the attention center and attention locality and then adapting the parameters during the training step. This work shows almost the same accuracy on ImageNet-1k as DeiT [11] using only 50% of the dataset, demonstrating the benefits of trying to mimic the inductive bias from CNNs. The architecture is trained based on the approach from DeiT [11] and ConViT-B has almost the same number of parameters in comparison to ViT-B.
- **GC ViT-B [46]:** This variant proposes a method to combine both the standard local context in self-attention with a global context, alternating both blocks to capture fine- and coarse-grained features. The global self-attention makes it possible to query image regions instead of patches (overlapping patches) by applying a convolutional layer. This work shows a greater accuracy on ImageNet-1k compared to ViT-B [10] with almost the same number of parameters.
- **MViTv2-B [47]:** The improved Multiscale Vision Transformers (MViTv2) architecture was proposed to work on both image and video domains. This architecture encodes relative position information in the self-attention and uses a pooling operation after the linear projection on both Q , K , and V inside the transformer block.
- **CaiT-S24 [48]:** This work proposed a method to make deeper ViT possible without saturating the accuracy. The proposed method divides the architecture into two stages, the self-attention and the class attention, where the first is identical to ViT, except that it has no class token, and the second integrates the patches into the class embedding and extracts more fine-grained patches for the class token, increasing the accuracy and making the training of deeper architectures viable. The CaiT-S24 architecture shows a greater accuracy on ImageNet-1k than ViT-B [10] with almost half of the parameters. Also, the training schedule was based on DeiT [11].
- **XCiT-M24/16 [49]:** This variant proposes a new self-attention mechanism that decreases the quadratic cost of the original approach. This is achieved by Cross-Covariance Attention, which operates across feature channels rather than tokens, resulting in a linear complexity in the number of tokens. This architecture is more efficient for processing high-resolution images and has a better scalability than the original ViT. The XCiT-M24/16 architecture used here was pre-trained using the DINO [41] self-supervised approach.
- **BeiT-B/16 [13]:** The Bidirectional Encoder representation from Image Transformers (BeiT) is a self-supervised approach that proposes masked image modeling to pre-train Vision Transformers, according to previous findings with the BERT architecture on large language models. It consists of learning to reconstruct image patches by randomly corrupting some original patches, and it can be applied to previous ViT variants. The B/16 variant corresponds to applying the BeiT self-supervised training framework to the ViT-B/16 architecture.

- **BeiTv2-B/16** [14]: This variant improves the previous BeiT self-supervised pre-training by using a semantic-rich visual tokenizer, achieved by vector-quantized knowledge distillation. This technique promotes masked image modeling from the pixel level to the semantic level, outperforming the previous approach on image classification and semantic segmentation tasks. The model used in this work was firstly pre-trained on ImageNet-1k using BeiT self-supervised training and then fine-tuned on ImageNet-21k using a supervised approach.
- **Swin-B** [50]: This variant introduces shifted windows between consecutive self-attention layers. This is achieved by a hierarchical/multi-stage architecture, where the input is firstly split into a common patch embedding (stage 1), and then patch merging layers are used as the depth is increased. For instance, the first patch merging layer (stage 2) concatenates the features of each group of 2×2 neighboring patches from the original patch embedding. The procedure is then repeated for the following stages using similar window merging approaches, decreasing the output resolution and resulting in a hierarchical representation structure. This approach also incurs linear computational complexity concerning image size, allowing the Swin architecture to show better compatibility with a broad range of vision tasks. The Swin-B model used in our experiments is pre-trained on the ImageNet-21k dataset in a supervised fashion.
- **Next-ViT-L** [51]: Next-ViT introduces a hybrid CNN–transformer backbone with deployment-friendly design blocks (the Next Convolution Block (NCB) and Next Transformer Block (NTB)) and a new stacking strategy (NHS) to balance efficiency and accuracy. The Large variant, pre-trained on ImageNet-1k, uses a hierarchical embedding dimension of 1024 and operates at 224×224 resolution with 10.7GFLOPs and 57.9M parameters.
- **SHViT-S4** [52]: The Single-Head Vision Transformer (SHViT) introduces a novel macro design that improves memory and computational efficiency. It replaces the standard multi-head attention mechanism with Single-Head Self-Attention (SHSA), which removes redundancy while maintaining accuracy. The architecture is composed of a hierarchical three-stage design using a patchify stem with 16×16 stride and combines depthwise convolutions and SHSA blocks. This configuration enables a reduction in latency compared to MobileViTv2.
- **ViT-SO400M/16-SigLIP2** [53,54]: This model combines the shape-optimized SoViT-400M [54] architecture with the SigLIP 2 contrastive vision–language training framework. The SoViT-400M backbone was obtained by scaling width, depth, and MLP dimensions to improve efficiency while maintaining performance, resulting in a configuration with approximately 400 million parameters, 27 layers, an embedding dimension of 1152, and an MLP dimension of 4304. In turn, SigLIP 2 enhances training by integrating contrastive learning with additional objectives such as captioning supervision, masked image modeling, and self-distillation, applied to the multilingual WebLI dataset. The ViT-SO400M/16-SigLIP2 model processes images of size 384×384 using a patch size of 16×16 .
- **NaFlexViT-B** [53]: NaFlexViT is a shape-flexible Vision Transformer architecture introduced in the SigLIP 2 framework to support inference on images with arbitrary resolutions and aspect ratios. The key architectural innovation lies in the ability to process non-uniform image shapes by avoiding fixed-size resizing or cropping while preserving model generalization and accuracy. It uses a 16×16 patch embedding stem and employs global average pooling (GAP) instead of a class token for final feature aggregation, making it resolution-agnostic. This model is trained on ImageNet-1k at a resolution of 384×384 , with approximately 86.6 million parameters and a compute cost of 55.9 GFLOPs.

Table 1. Taxonomy of the ViT variants used in this work and the baselines considered for comparison. We indicate the pre-training that was used by the variants employed in this work (we used ResNet50 and DeiT3 with both IN-1k and IN-21k versions), where $a \Rightarrow b$ means that the model was pre-trained on dataset a , fine-tuned on dataset b , and then used for feature extraction. The feature extraction costs consider a 224×224 RGB input, and the size d indicates the dimensionality of the feature vector. For ViTs and ResNet50, the GFLOPs and the number of parameters refer only to the model backbone used for feature extraction, i.e., after removing the classification head. Models with * indicate 384×384 RGB input.

Model	Embedding	Pre-Training		d	Feature Extraction Cost	
		Dataset	Paradigm		GFLOPs	Param. (m)
hand-eng. baseline (LBP)	N.A.	N.A.	N.A.	≈ 256	≈ 0.05	0
CNN baseline (ResNet50)	convolutional	ImageNet-1k and 21k	supervised	2048	4.1	25.5
CoaT-Li-Mi	convolutional	ImageNet-1k	supervised	512	2.0	10.5
CoaT-Mi	convolutional	ImageNet-1k	supervised	216	7.2	10.1
MobileViT-S	convolutional	ImageNet-1k	supervised	640	1.4	4.9
MobileViTv2	convolutional	ImageNet-1k	supervised	512	1.4	4.4
EfficientFormer-L1	patches	ImageNet-1k	supervised	448	1.3	11.4
EfficientFormer-L3	patches	ImageNet-1k	supervised	512	3.9	30.4
ViT-B/16	patches	ImageNet-21k	supervised	768	16.9	85.8
ViT-B/16-DINO	patches	ImageNet-1k	self-supervised	768	16.9	85.8
ViT-B/16-SAM	patches	ImageNet-1k	supervised	768	16.9	85.8
DeiT-B/16	patches	ImageNet-1k	supervised	768	16.9	85.8
DeiT3-B/16	patches	ImageNet-1k and 21k	supervised	768	16.9	85.8
CrossViT-B	patches	ImageNet-1k	supervised	1152	20.1	103.9
ConViT	convolutional	ImageNet-1k	supervised	768	16.8	85.8
GC ViT-B	convolutional	ImageNet-1k	supervised	1024	13.9	89.3
MViTv2-B	patches	ImageNet-1k	supervised	768	8.9	50.7
CaiT-S24	patches	ImageNet-1k	supervised	384	8.6	46.5
XCiT-M24/16	patches	ImageNet-1k	self-supervised	512	15.8	83.9
BeiT-B/16	patches	ImageNet-21k	self-supervised	768	12.7	85.8
BeiT2-B/16	patches	ImageNet-1k \Rightarrow 21k	self-sup. \Rightarrow sup.	768	12.7	85.8
Swin-B	patches	ImageNet-21k	supervised	1024	15.1	86.7
ViT-SO400M/16-SigLIP2 *	patches	WebLI	self-supervised	1152	~ 35.0	428
SHViT-S4	convolutional	ImageNet-1k	supervised	768	6.5	28.2
Next-ViT-L	convolutional	ImageNet-1k	supervised	1024	10.7	57.9
NaFlexViT-B *	patches	ImageNet-1k	supervised	768	55.9	86.6

3. Methodology

3.1. Vision Transformer's Features Off the Shelf

A common approach to employ foundation models for a novel task, especially in data-scarce scenarios, is to remove the classification head, freeze the pre-trained backbone, and then train only a linear classifier over their “features off the shelf”. In CNNs, this can be achieved by applying global average pooling after convolutional layers or considering the output of fully connected layers. The features of the latter approach, however, are known to be highly correlated with the spatial order of the pixels [25].

In most of the ViT architectures, the output of the penultimate layer, i.e., the class token, or x_L^0 (see Equation (3)), is already suited as an image representation without any additional transformation, since it is a 1-dimensional embedding vector. Moreover, the relation of this embedding with the spatial order of the pixels is not direct as in CNNs. In some cases, such as for the Swin architecture [50], a global average pooling operator is applied over the output feature map of the last transformer layer to obtain the image representation instead of using the class token. In any case, our goal is to analyze how these representations behave for texture analysis. In this context, the ViT itself is not fine-tuned, and the base architecture (backbone) is not modified (except for removing the original

output head). This allows us to analyze the potential of existing foundation models when directly applied to texture recognition tasks.

3.2. Linear Classifiers

The image representations obtained with ViT backbones are used to train three classifiers, including linear ones such as LDA and SVM, as well as the non-linear but simple KNN classifier. We focus on classifiers that can be trained with less data compared to deep learning models and that have been studied and used for several decades. Their hyperparameters are also not tuned, as the focus of this work is on evaluating the quality of the features extracted by ViTs. The following supervised classifiers from the Scikit-learn [55] (version 1.6.0) Python library are considered:

- **KNN:** k -Nearest Neighbors [56] using $k = 1$;
- **LDA:** Linear Discriminant Analysis [57], with the least-squares solver and automatic shrinkage using the Ledoit–Wolf lemma;
- **SVM:** Support Vector Machine [58], with a linear kernel and $C = 1$.

After fitting each classifier, individually, over the features extracted with a ViT backbone using the image training set, we evaluate the performance on the corresponding test set. It is important to note that the training and test split protocols follow the standard procedures recommended in the reference papers for each dataset and are described in detail in Section 4.1.2. The classification accuracy for each classifier is computed as the ratio of correct predictions to the total number of test samples. We then report the average of the three accuracy values (one per classifier) as the final result. This average classification accuracy provides a single performance metric that reflects the mean accuracy across the three classifiers. This approach is employed to minimize the variance caused by the different classification paradigms; e.g., the features of some ViTs may be better coupled with a specific classifier. For instance, in our experiments, while LDA and SVM performed better in general, the KNN classifier surpassed them in some cases. Therefore, we believe that the average results of the three different classifiers should provide a better overall estimate of the quality of the ViT features in different scenarios.

4. Experiments and Results

4.1. Experimental Setup

Following the methodology described above, we evaluate the ViT variants using an Ubuntu server with two Nvidia GeForce GTX 1080ti graphic cards (11 GB of VRAM each), an Intel Core i7-7820X processor, and 64 GB of RAM. The scripts are implemented using PyTorch [59] (version 2.1.0). We employ the PyTorch Image Models library [60] (also known as timm, using Version 0.6.7) to obtain both the model implementation and pre-trained weights, since this is a widely used library in the computer vision community. The ViT feature vectors obtained with timm are then employed for the classification step using the supervised classifiers. We evaluate the performance in terms of the average classification accuracy among the three aforementioned linear classifiers (KNN, LDA, and SVM).

4.1.1. Baselines for Comparison

Aside from the ViTs, two additional approaches are used in our experiments as a baseline for comparison. Following the developments in the texture analysis field, we consider the classic LBP [19] method, which was the predominant approach in many computer vision applications before the proliferation of deep learning. While most of the hand-engineered baseline results in this paper refer to LBP, we also include some results using Gabor filters [20], 3-D RGB histograms, and Improved Fisher Vectors (IFVs) [61].

An important difference between hand-engineered and deep learning approaches is their computational cost. For instance, the original implementation of the LBP method has an $O(rn)$ time complexity, where n is the number of grayscale pixels in an image and r is the size of the analyzed neighborhood. In our case, $n = 150,528$ for a 224×224 RGB image, and r is usually between 8 and 24, which yields around 1.2 to 3.6 million operations (around 0.001 to 0.004 GFLOPs). A common approach to achieving multi-resolution grayscale and rotation-invariant LBP descriptors, as in ref.[19], is to combine different neighborhood sizes (e.g., 8, 16, and 23), and bin sizes for computing the local binary pattern histogram. We will assume that the cost of LBP is ≈ 0.05 GFLOPs (and ≈ 256 features, since this number may vary depending on the parameters), i.e., higher than combining 10 neighborhoods of size 23, which is an overestimate to account for its many possible use cases. Nevertheless, this cost consists of only a fraction of that of deep learning-based models. Hand-engineered techniques may also benefit from current hardware; e.g., the LBP method may reach a $\theta(1)$ time complexity with recent parallel implementations for GPUs [62].

We also consider the ResNet50 architecture [63] as a CNN baseline, which is one of the most frequently used convolutional models. This CNN is pre-trained on IN-1k according to the original source. Additionally, we consider another version of ResNet50 that uses knowledge distillation and IN-21k pre-training [64]. In terms of cost, compared to the base versions of most ViTs, ResNet50 has a lower computational budget but higher feature dimensionality (see Table 1). Further on, we address different aspects of the computational cost in our efficiency analysis (see Section 4.3).

4.1.2. Texture Recognition Tasks

Eight texture datasets are considered in this work in order to analyze a variety of scenarios. They cover several texture recognition tasks such as the classification of materials and texture instances, as well as related properties such as robustness to image transformations. The task difficulty ranges from homogeneous texture images acquired under controlled settings to datasets with a variety of textures taken from the Internet. The evaluation policy (training/test splits) also varies among them. We describe each dataset below (Figure 3 shows some samples for each one):

- **Outex10** [65]: This dataset consists of 4320 grayscale images belonging to 24 different textures classes, where the train split contains 480 images and the test split contains 3840 images. The same textures are rotated at nine different angles (0, 5, 10, 15, 30, 45, 60, 75, and 90).
- **Outex11** [65]: This dataset consists of 960 grayscale images representing 24 different texture classes, where the train split is composed of 480 images and the test split contains 480 images. This dataset represents textures under different scales.
- **Outex12** [65]: This version is composed of 9120 grayscale images representing 24 different textures, which is split into two folds, where each fold has the same 480 images in the train split and 4320 test images (two test folds). This dataset represents textures under nine different rotation angles and different illumination.
- **Outex13** [65]: This dataset is composed of 1360 RGB images of 68 texture classes and evaluates color texture recognition. The samples are split into 680 images for training and 680 images for testing.
- **Outex14** [65]: This dataset contains 4080 RGB images corresponding to 68 texture classes and evaluates color texture recognition under different illumination. The train split contains 680 images, while the test split contains 1360 images.
- **Describable Texture Dataset (DTD)** [25]: This dataset is composed of 5640 images belonging to 47 texture classes, with images taken from the Internet with minimal

control (textures in the wild). It is evaluated on the 10 provided splits for training, validation, and test.

- **Flickr Material Dataset (FMD)** [66]: This dataset holds 1000 images representing 10 material categories, also obtained from the Internet. The validation is conducted through 10 repetitions of 10-fold cross-validation.
- **KTH-TIPS2-b** [67]: This dataset contains 4752 images of 11 different materials, which are split according to a fixed four-fold cross-validation. The images have nine different scales equally spaced logarithmically per sample, three camera poses (frontal, 22.5° left, and 22.5° right), and four illumination conditions (front, from the side at roughly 45°, from the top at roughly 45°, and using ambient lighting).

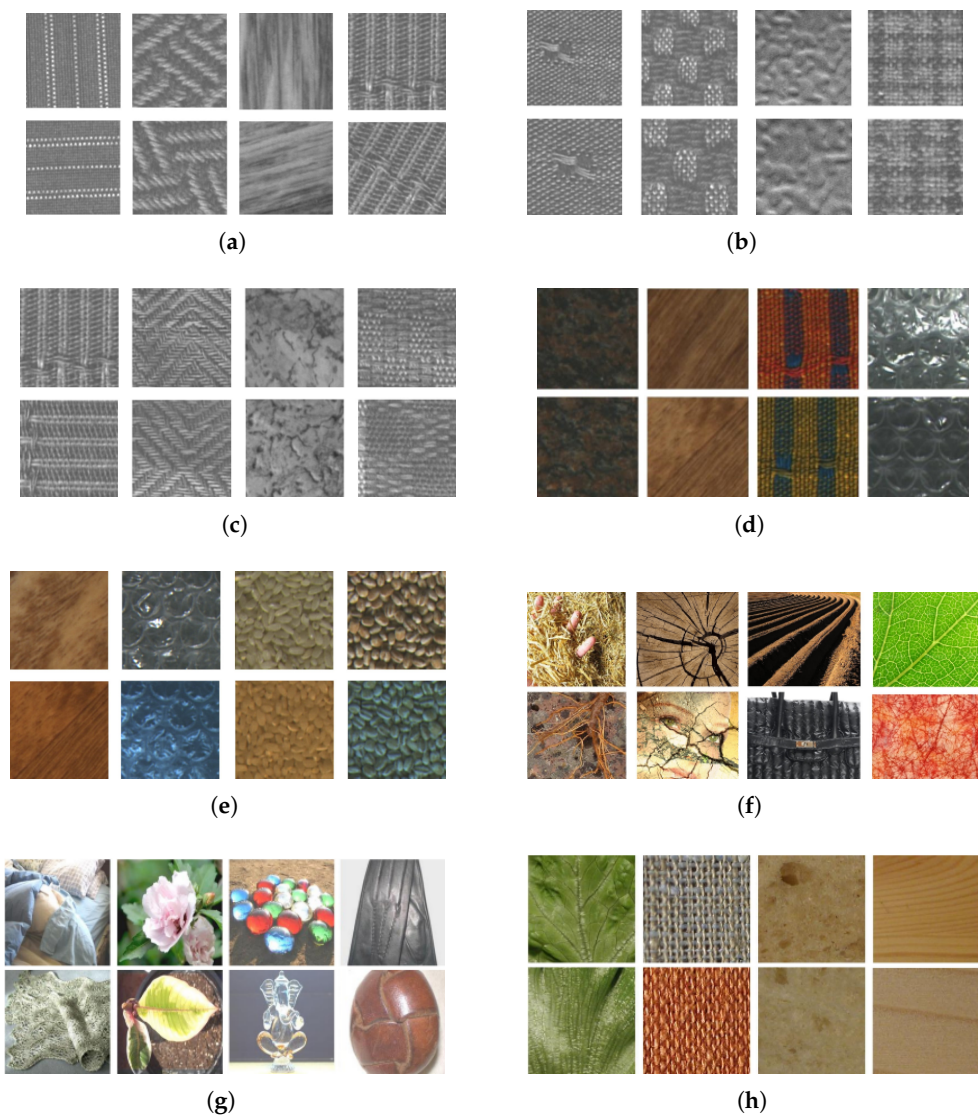


Figure 3. Texture samples from the eight image datasets used in this work. For each dataset, each column represents a different texture class, while each row represent different samples from that class. (a) Outex10. (b) Outex11. (c) Outex12. (d) Outex13. (e) Outex14. (f) DTD. (g) FMD. (h) KTH-2-b.

4.2. Performance Comparison

Our first analysis deals with a general comparison of performance across different texture recognition tasks. These metrics directly reflect the quality of the feature vectors, or image representations, that can be obtained with pre-trained foundation ViTs with open-source code and weights. We divide the texture recognition tasks into two groups:

(a) theoretical robustness analyses and (b) more complex and realistic tasks. We present the results and corresponding discussion for each group in the following.

4.2.1. Robustness to Geometric Transformations and Illumination

We select the Outex10, Outex11, and Outex12 datasets for evaluating the robustness of ViTs. As previously described, these datasets are designed to evaluate the performance of texture recognition models under rotation, scale, and illumination changes. The results for the hand-engineered and CNN baselines and all ViTs are shown in Table 2. It represents the average classification accuracy of the three linear classifiers after being independently trained and validated according to each dataset cross-validation split. The results are highlighted compared with the baselines and the best results obtained on each dataset. The table is also divided into blocks of rows according to the different approaches for feature extraction: baselines, mobile ViTs, and base ViT models with IN-1k or IN-21k pre-training.

Table 2. Average classification accuracy of three linear classifiers (KNN, LDA, and SVM) learned over the output features of each ViT backbone. We also list CNN baselines with ResNet50 (with IN-1k and IN-21k pre-training) and the hand-engineered baseline results from the Outex (2002) [65] paper, based on LBP and Gabor descriptors using the best among a variety of classifiers. **Bold type** indicates results above the hand-engineered baseline results, and **blue** indicates results above the CNN baseline results (according to pre-training).

Model	Outex10	Outex11	Outex12
hand-engineered (LBP)	97.9	99.2	87.2
CNN baseline (ResNet50)	85.1 \pm 1.5	99.8 \pm 0.2	87.7 \pm 1.4
CoaT-Li-Mi	82.6 \pm 0.7	99.4\pm0.2	84.4 \pm 0.6
CoaT-Mi	85.1 \pm 0.9	99.3\pm0.2	88.1\pm0.9
MobileViT-S	70.1 \pm 0.4	87.4 \pm 3.7	62.0 \pm 3.0
MobileViTv2	69.3 \pm 0.9	95.8 \pm 2.2	69.5 \pm 0.8
EfficientFormer-L1	86.0\pm0.1	99.8\pm0.3	86.4 \pm 0.4
EfficientFormer-L3	91.3\pm0.7	100\pm0.0	92.1\pm0.7
ViT-B/16-DINO	93.5\pm1.3	100\pm0.0	94.0\pm1.1
DeiT-B/16	91.9\pm0.6	99.9\pm0.1	92.5\pm0.7
DeiT3-B/16	86.1\pm1.3	99.9\pm0.1	88.0\pm1.6
ViT-B/16-SAM	90.9\pm1.2	100\pm0.0	91.8\pm0.8
CrossViT-B	87.4\pm1.5	99.2 \pm 0.3	89.1\pm1.5
ConViT-B	86.8\pm0.2	100\pm0.0	88.8\pm0.2
GC ViT-B	92.5\pm0.7	99.9\pm0.1	91.8\pm0.7
MViTv2-B	91.2\pm0.6	99.7\pm0.1	92.3\pm0.4
CaiT-S24	91.3\pm0.9	99.9\pm0.2	91.5\pm0.8
XCiT-M24/16	87.7\pm0.1	99.0 \pm 0.5	89.7\pm0.6
CNN baseline (ResNet50 IN-21k)	96.9 \pm 1.1	99.5 \pm 0.3	96.9 \pm 1.1
ViT-B/16 (IN-21k)	98.4\pm0.5	99.8\pm0.2	97.2\pm0.5
DeiT3-B/16 (IN-21k)	84.1 \pm 1.3	99.6\pm0.0	85.7 \pm 1.0
BeiT-B/16 (IN-21k)	98.4\pm0.4	100\pm0.0	97.8\pm0.3
BeiTv2-B/16 (IN-21k)	96.5 \pm 0.3	100\pm0.0	97.1\pm0.3
Swin-B (IN-21k)	97.8\pm0.6	100\pm0.0	97.0\pm0.3

On the Outex10 dataset, ResNet50 is outperformed by the majority of the ViTs, which was expected given the limitation of CNNs regarding global patterns and long-range dependencies, which are crucial for analyzing rotated texture images (see examples from this dataset in Figure 3a). On the other hand, only two ViT models additionally outperformed the hand-engineered baselines, namely ViT-B/16 (IN-21k) and BeiT-B/16 (IN-21k). These results reflect the data-hungry aspects of ViTs, where IN-1k pre-training does not suffice to

achieve better rotation robustness than classical methods such as LBP. Additionally, they also show that ResNet50 is not able to outperform the hand-engineered baseline (LBP) even when using IN-21k pre-training, highlighting a rotation robustness deficit of ImageNet pre-trained models when dealing with controlled images (or homogeneous texture images).

On the Outex11 and Outex12 datasets, the majority of the ViTs outperform both the hand-engineered and CNN baselines, except for the mobile ViTs. Many of the base-sized ViTs exhibit a high robustness (above 99.5% accuracy) to the changes in texture scales that are present in Outex11. As for Outex12, which contains illumination changes and is a harder task for the baselines, many ViTs also outperform them. In this sense, this result shows that pre-trained ViTs (of base size) using either IN-1k or IN-21k pre-training achieve a strong scale and illumination robustness compared to the baselines. Some small ViTs, such as CoaT-Mi and EfficientFormer-L1, may also be viable in this scenario given their lower computational budgets, even though they do not surpass the baseline performance in all cases. Particularly, it is worth highlighting the robustness to illumination changes of ViT models with IN-21k pre-training specifically (except DeiT-B/16), which outperform the baselines and IN-1k pre-training with a considerable margin.

It is also worth pointing to the degraded performance of some models in this first evaluation step, especially MobileViT-S and DeiT-B/16. This is related to their higher sensitivity to rotation, scale, and illumination changes in textures. However, the representation obtained from the last transformer layer may not be ideal for this application (homogeneous textures), considering the complexity of these features. In this sense, multi-depth feature engineering and aggregation may be necessary for improving the ViTs in these cases to benefit from earlier features, as has been performed with pre-trained CNNs when applied to texture analysis [26].

4.2.2. Complex and In-the-Wild Texture Recognition

The second evaluation step focuses on texture recognition datasets representing more challenging scenarios. The datasets may contain variations in rotation, scale, and illumination like the previous datasets, but the classification tasks become harder due to a series of other factors. We consider five datasets: Outex13 (color textures), Outex14 (color textures with illumination changes), DTD (texture attributes in the wild), FMD (materials in the wild), and KTH-TIPS2-b (materials under several conditions). The results for all ViTs and baselines are shown in Table 3. In this comparison, we also include recent literature methods that are based on mathematical models and CNN backbones: RADAM (ResNet50) [26], Multilevel Pooling [68], DTPNet [69], and Fractal Pooling [70].

The results for the Outex13 and Outex14 datasets, which evaluate the ability to deal with color textures, show that while many ViTs can outperform ResNet50 and RADAM, just a few of them outperform the hand-engineered baselines. For instance, on Outex13 no neural network was able to surpass the results obtained with a 3-D RGB histogram (from ref.[65]), which is a considerably simpler method. As for Outex14, which focuses specifically on illumination changes in color textures, the following methods outperform both baselines: ViT-B/16-DINO and DeiT3 on IN-1k and using IN-21k pre-training with ViT-B/16, DeiT3-B/16, and BeiTv2-B/16. Furthermore, we highlight the results of ViT-B/16-DINO, which achieves the highest performance among the ViTs on these two datasets and represents a considerable improvement on Outex14 (78.4% versus 69% from LBP). In conclusion, although hand-engineered baselines are strong candidates for characterizing color textures, some architectures and improved pre-training approaches for ViTs may be better in some cases.

Table 3. Average classification accuracy considering more challenging texture recognition tasks involving color, different materials, and patterns collected “in the wild”. The hand-engineered results for Outex are from ref.[65], using a 3-D RGB histogram (Outex13) and LBP (Outex14). The hand-engineered results for DTD and FMD, based on the IFV method, are from ref.[25], and the KTH results using LBP are from ref.[67]. Methods marked with * have their results taken from the original paper, while dashes (–) indicate unavailable results for those datasets. **Bold type** indicates results above the hand-engineered baseline results, and **blue** indicates results above the CNN baseline results (according to pre-training).

Method	Outex13	Outex14	DTD	FMD	KTH-2-b
hand-engineered (LBP)	94.7	69.0	61.2	58.2	84.0
CNN baseline (ResNet50)	87.6 \pm 2.2	54.4 \pm 0.7	69.2 \pm 2.9	81.8 \pm 3.9	84.6 \pm 1.3
RADAM (ResNet50) * [26]	90.7 \pm 2.6	63.6 \pm 0.6	73.1 \pm 3.0	81.3 \pm 6.5	86.8 \pm 1.4
Multilevel Pooling * [68]	–	–	83.1 \pm 0.3	–	93.4 \pm 3.6
DTPNet * [69]	–	–	73.5 \pm 0.4	87.8 \pm 1.3	88.5 \pm 1.6
Fractal Pooling * [70]	–	–	–	89.3	91.2
ViT model					
CoaT-Li-Mi	85.8 \pm 3.3	60.6\pm2.2	66.3\pm4.0	79.9\pm5.7	88.0\pm2.6
CoaT-Mi	85.0 \pm 2.4	57.7\pm2.8	64.0\pm2.2	79.6\pm4.5	84.5\pm2.8
MobileViT-S	85.0 \pm 1.8	23.6 \pm 0.8	19.4 \pm 2.6	27.1 \pm 4.1	56.8 \pm 1.7
MobileViTv2	74.4 \pm 2.6	31.6 \pm 3.9	18.0 \pm 1.2	25.2 \pm 3.2	58.3 \pm 3.1
EfficientFormer-L1	90.0\pm2.4	64.6\pm1.5	70.0\pm2.6	83.4\pm3.7	86.9\pm1.7
EfficientFormer-L3	89.6\pm2.1	64.2\pm1.4	70.7\pm3.1	83.9\pm4.1	86.2\pm1.0
ViT-B/16-DINO	94.2\pm1.3	78.4\pm0.9	74.0\pm2.2	85.2\pm3.8	88.6\pm1.0
DeiT-B/16	90.8\pm2.5	65.8\pm1.2	67.2\pm5.4	79.9\pm7.6	87.4\pm1.7
DeiT3-B/16	87.9\pm3.1	70.0\pm3.9	67.2\pm4.4	83.2\pm4.4	86.1\pm3.1
ViT-B/16-SAM	92.6\pm1.8	65.6\pm1.2	65.5\pm4.2	77.7\pm4.9	80.8\pm1.5
CrossViT-B	88.8\pm2.9	63.6\pm3.1	63.6\pm5.4	77.5\pm5.4	87.9\pm2.4
ConViT-B	89.5\pm2.5	67.9\pm2.2	66.8\pm5.4	80.6\pm9.9	86.1\pm1.8
GC ViT-B	89.9\pm2.8	67.4\pm3.0	70.1\pm4.6	85.6\pm5.6	82.1\pm2.5
MViTv2-B	87.4\pm3.4	65.3\pm1.7	69.1\pm4.5	83.3\pm4.5	87.1\pm1.5
CaiT-S24	90.4\pm3.0	67.1\pm1.6	68.3\pm3.0	82.6\pm5.0	88.1\pm1.7
XCiT-M24/16	88.7\pm3.4	62.3\pm1.0	62.2\pm5.3	78.4\pm5.8	86.4\pm2.8
ViT-SO400M/16-SigLIP2	89.7\pm2.2	49.9\pm3.4	61.0\pm5.7	55.3\pm8.9	81.8\pm2.0
SHViT-S4	85.8\pm2.3	60.3\pm2.9	63.5\pm4.8	76.5\pm6.5	84.7\pm2.9
Next-ViT-L	91.2\pm2.4	64.0\pm1.5	72.6\pm3.8	84.6\pm5.9	84.4\pm1.3
NaFlexViT-B	90.7\pm2.7	66.1\pm1.4	65.9\pm5.4	76.4\pm7.6	85.3\pm1.3
CNN baseline (ResNet50 IN-21k)	90.5 \pm 1.3	66.3 \pm 0.5	74.0 \pm 1.9	86.1 \pm 4.1	84.7 \pm 0.4
ViT-B/16 (IN-21k)	92.2\pm2.4	71.0\pm3.1	71.0\pm3.6	82.3\pm6.0	86.5\pm1.3
DeiT3-B/16 (IN-21k)	88.8 \pm 3.2	71.6\pm3.2	70.1\pm3.9	83.9\pm4.9	89.3\pm1.2
BeiT-B/16 (IN-21k)	89.0 \pm 1.5	43.3 \pm 1.5	47.9 \pm 4.4	60.0\pm9.3	78.6 \pm 1.9
BeiTv2-B/16 (IN-21k)	91.6\pm1.8	73.3\pm1.1	79.1\pm2.8	90.9\pm4.6	93.7\pm1.2
Swin-B (IN-21k)	89.8 \pm 2.5	68.6\pm1.3	78.6\pm2.9	90.5\pm5.4	88.4\pm1.0

DTD and FMD are datasets with texture images obtained in the wild (from the Internet). In this case, the texture recognition task is considerably more challenging, since models have to deal with a wide variety of scenarios, noise, multiple objects, conditions, backgrounds, etc. In this sense, the models need to deal not only with texture recognition but also with object detection. In these cases, hand-engineered methods struggle in comparison to neural networks, as their performance shows. Nevertheless, the hand-engineered baseline can outperform or perform similarly to some of the compared ViTs, especially some small/mobile architectures. In general, most of the base-sized ViTs outperform the hand-engineered baseline, and some of them are also able to outperform ResNet50. When compared to other recent methods from the literature, many base-sized ViTs also achieve superior results, outperforming models such as DTPNet, Fractal Pooling, and RADAM. The only exception is Multilevel Pooling, which achieves the highest accuracy on the DTD dataset. We again point to models with IN-21k pre-training, where the gains can be

expressive. For these tasks, we highlight EfficientFormer as a strong small-scale architecture and BeiTv2 and Swin as the best-performing alternatives.

The last dataset analyzed here is KTH-TIPS2-b. Compared to the previous ones, this dataset has different properties that deserve special attention. Firstly, the images are collected in a controlled setting, and textures cover all their area (see Figure 3h). The textures contain several variations such as scale, view angle, and illumination condition. This means that there is no need for background removal, object detection, and similar skills at which deep learning models excel, and the goal is solely to discriminate the target textures. In this sense, this dataset combines various transformations analyzed before and is crucial for comparing the capabilities of different approaches. The results show that most methods obtain a similar classification accuracy in the range 84% to 89%. However, most of the ViTs outperform the hand-engineered and CNN baselines. The literature approaches Multilevel Pooling and Fractal Pooling achieved competitive accuracies, but their performance was inferior to the BeiTv2-B/16 (IN-21k) model. This reflects their potential for texture analysis, corroborating that some ViTs can be powerful alternatives for hand-engineered methods and CNNs.

Another important aspect is the differences in architectural design and training schedules among the ViTs. Although most models are considerably similar to the standard ViT in terms of architecture, some differences such as different embedding approaches or self-attention mechanisms may be related to their performance in texture recognition. Moreover, the way the models are pre-trained may also be critical. Firstly, we observe that models with patch embedding tend to perform generally better than convolutional embeddings. This may be related to the fact that most mobile ViTs use convolutional embeddings, which is understandable considering their focus on a lower computational budget. However, EfficientFormer is situated among the mobile models but uses patch embedding and generally achieves a considerably higher performance than the other mobile variants, which supports our claim about the superiority of patch embeddings for texture recognition. In terms of the architecture, we note that although the common ViT-B architecture performs well, some variants with different mechanisms such as the Swin transformer (which uses shifted windows) achieve a superior performance.

Considering the pre-training differences among the ViTs, aside from the obvious difference between IN-1k and IN-21k, we note that self-supervised approaches (DINO and BeiT) generally perform better than supervised approaches. However, the DINO approach works considerably better with the basic ViT-B architecture than with the XCiT architecture, suggesting that feature channel attention may not be ideal for textures compared to token attention. BeiT, another self-supervised approach that employs masked image modeling, is among the best methods for homogeneous texture images and basic transformations but struggles with more complex and in-the-wild texture images. Nevertheless, this seems to have improved for BeiTv2, which enhanced masked image modeling from the pixel to the semantic level, making it perform on par with the best methods and also have the best performance for complex and in-the-wild textures.

4.3. Efficiency Analysis

Performance is not the only desirable trait of CV models. Indeed, efficiency is another strong aspect, especially when considering low-cost hardware or mobile devices. Therefore, in this section, we discuss the efficiency of the ViTs in terms of feature extraction cost. We removed MobileViT-S and MobileViTv2 from this analysis for better visualization, considering their degraded performance observed in the previous experiments.

4.3.1. Computational Complexity

The computational cost of neural networks or other machine learning models can be estimated using a variety of properties. We consider three measurements that are commonly employed in the deep learning and computer vision literature: the number of floating point operations (FLOPs), the number of parameters, and the number of features (feature vector size). The combination of these measurements provides a consistent estimation of the feature extraction cost for the ViTs under comparison. For instance, the number of FLOPs estimates the processing time, the number of parameters estimates the memory consumption, and the number of features shows the size needed to encode the images (feature dimensionality) as well as the cost of using these features for pattern recognition (classification, regression, etc.). Figure 4 shows the results of this analysis, where we consider the correlation between the cost measurements and the average performance in the texture recognition tasks.

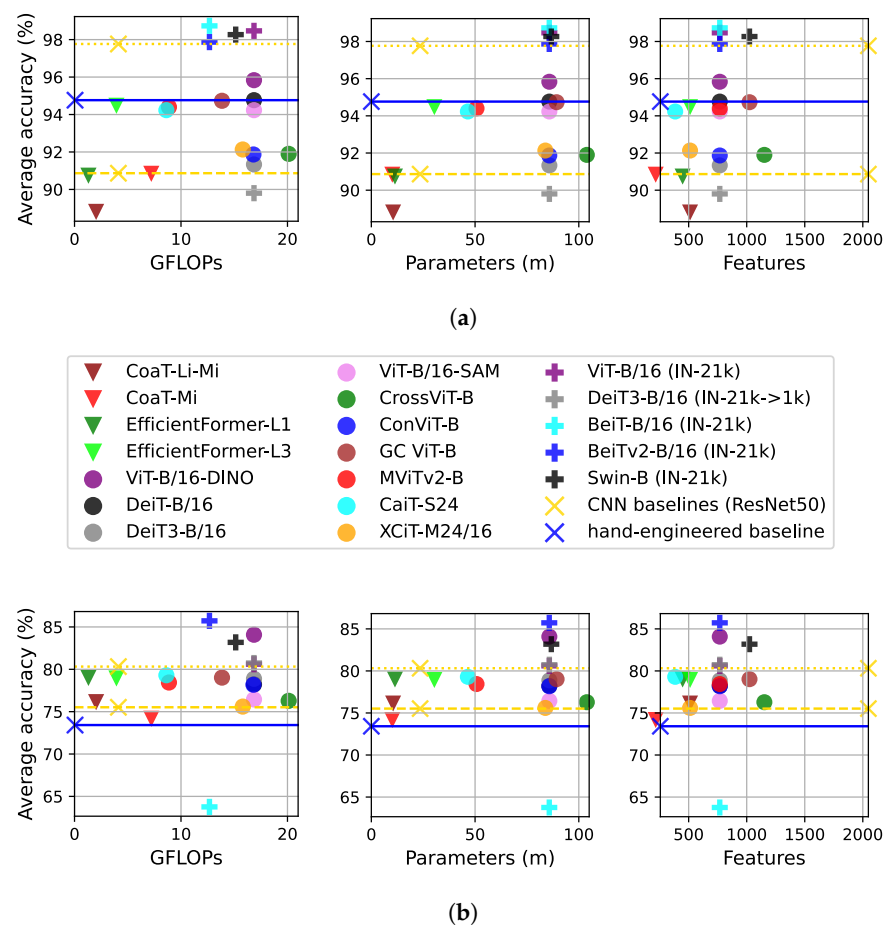


Figure 4. Efficiency analysis of ViT variants compared to hand-engineered and CNN baselines, where accuracy represents the average accuracy over the corresponding datasets and classifiers considered (KNN, LDA, and SVM). The yellow line with the smaller dots represents ResNet50 with IN-21k pre-training. (a) Average accuracy on Outex 10, 11, and 12. (b) Average accuracy on Outex13, 14, DTD, FMD, and KTH-2-b.

Figure 4a considers the average performance on the first three datasets (Outex 10, 11, and 12, see also Table 2). These results indicate the correlation between cost and robustness to rotation, scale, and illumination. The few ViTs that outperform the baselines have a significantly higher computational cost, except in terms of number of features, where ViTs use a smaller image representation than ResNet50. The best alternative is the

EfficientFormer-L3, but this ViT variant is outperformed by ResNet50 with IN-21k pre-training. This situation may change with stronger pre-training of mobile or other smaller ViT variants to improve their robustness.

Figure 4b focuses on datasets representing a more challenging scenario (Outex13 and 14, DTD, FMD, and KTH-2-b, see also Table 3). In general, ViTs performs better on these more complex tasks, but the highest performance achieved by the bigger variants comes with a considerably higher computational budget than that of ResNet50. Nevertheless, the EfficientFormers and Coat-Li-Mi architectures arise as powerful alternatives in this scenario, offering a balance between efficiency and performance compared to the baselines.

4.3.2. Feature Extraction Running Time

In practice, the computational budget of deep neural networks may also depend on the quality of implementation, code optimization, hardware, etc. In this sense, we performed an additional experiment to measure the real running time of each ViT when used for feature extraction on one or more images (batches), also referred to as model throughput (images processed per second). We consider running the models using either CPU or GPU processing and also by varying the batch size (1, 8, or 16). The results for this experiment are shown in Table 4. For each case (cell of the table), we run 100 repetitions and compute the average and standard deviation of the processing times.

Table 4. Throughput (images per second, the higher the better) of models performing feature extraction (average of 100 repetitions) using batches of 224×224 RGB images, performed on a machine with a GTX 1080ti, Intel Core i7-7820X 3.60 GHz processor, and 64GB of RAM. Cells in blue represent methods with a throughput higher than ResNet50 (CNN baseline) in the respective column.

Model	Batch Size (CPU)			Batch Size (GPU)		
	1	8	16	1	8	16
CNN baseline (ResNet50)	44.59 \pm 5.35	113.12 \pm 9.86	42.52 \pm 1.23	158.43 \pm 3.30	1226.90 \pm 22.17	2494.15 \pm 44.70
Coat-Li-Mi	48.06 \pm 2.16	141.89 \pm 11.35	58.60 \pm 2.01	208.83 \pm 8.53	672.77 \pm 24.71	1352.99 \pm 24.88
Coat-Mi	15.83 \pm 0.66	47.09 \pm 2.23	21.84 \pm 0.51	68.65 \pm 1.22	180.24 \pm 4.01	109.42 \pm 0.21
MobileViT-S	49.39 \pm 3.10	128.42 \pm 7.43	43.46 \pm 1.73	146.92 \pm 3.10	1193.77 \pm 23.76	2375.24 \pm 53.52
MobileViTv2	52.90 \pm 3.19	148.75 \pm 7.44	49.31 \pm 1.39	122.57 \pm 2.11	974.70 \pm 15.99	1854.68 \pm 32.93
EfficientFormer-L1	67.88 \pm 6.27	218.89 \pm 21.47	83.22 \pm 2.40	169.66 \pm 6.63	1323.82 \pm 30.80	2628.99 \pm 90.08
EfficientFormer-L3	33.34 \pm 2.05	97.24 \pm 4.97	37.02 \pm 1.21	105.00 \pm 3.46	812.24 \pm 14.43	1646.44 \pm 36.23
ViT-B/16	20.38 \pm 1.47	46.34 \pm 2.26	21.99 \pm 0.91	302.58 \pm 11.11	2315.91 \pm 56.24	4937.86 \pm 111.52
DeiT-B/16	20.26 \pm 1.58	46.10 \pm 2.25	21.45 \pm 0.77	311.27 \pm 13.23	2324.72 \pm 46.65	4979.28 \pm 104.35
DeiT3-B/16	19.99 \pm 1.61	46.16 \pm 2.15	20.13 \pm 0.62	286.87 \pm 17.90	2154.10 \pm 38.75	4586.07 \pm 95.42
CrossViT-B	13.11 \pm 1.28	35.30 \pm 1.41	15.55 \pm 0.33	157.73 \pm 5.22	1173.96 \pm 28.97	2459.55 \pm 50.21
ConViT-B	14.71 \pm 0.98	32.49 \pm 1.08	14.75 \pm 0.22	190.96 \pm 2.42	1448.37 \pm 17.13	3189.61 \pm 278.31
GC ViT-B	10.03 \pm 0.73	25.64 \pm 0.92	11.44 \pm 0.70	65.99 \pm 3.16	236.71 \pm 4.92	104.65 \pm 0.82
MViTv2-B	13.30 \pm 0.51	39.50 \pm 1.34	17.48 \pm 0.28	50.38 \pm 0.74	185.04 \pm 0.42	106.18 \pm 0.19
CaiT-S24	17.69 \pm 1.06	46.14 \pm 1.76	23.60 \pm 0.51	108.59 \pm 3.86	461.23 \pm 11.32	182.48 \pm 0.56
XCiT-M24/16	12.71 \pm 0.68	35.53 \pm 0.92	18.59 \pm 0.33	74.12 \pm 1.75	240.27 \pm 4.87	133.77 \pm 1.06
BeiT-B/16	17.99 \pm 1.05	44.27 \pm 1.74	20.48 \pm 0.31	235.56 \pm 7.69	1750.17 \pm 32.89	3753.43 \pm 52.67
BeiT2-B/16	18.51 \pm 0.33	45.70 \pm 0.13	20.53 \pm 0.11	233.99 \pm 4.65	1786.92 \pm 19.56	3740.98 \pm 48.88
Swin-B	11.41 \pm 0.13	30.74 \pm 0.07	16.64 \pm 0.14	104.62 \pm 1.26	819.50 \pm 10.15	292.53 \pm 5.43

We observe that the ViT throughput, in practice, is more nuanced than their performance or estimated cost. Firstly, it is important to stress that the throughput decreases when increasing the batch size from 8 to 16 on the CPU due to the fact that this is an 8-core/16-thread processor. The results also show that mobile or low-cost architectures are generally faster than the CNN baseline (ResNet50) when running on the CPU, while the situation changes on the GPU, where only the EfficientFormer-L1 outperforms it. On

the other hand, considering the larger ViTs, none of them are faster than ResNet50 on the CPU, while some of them can be up to two times faster than the CNN on the GPU. We highlight the ViT-B and DeiT architectures (which are similar but have different codes), which achieve the highest throughputs while running on the GPU.

The nuances in efficiency can be explained by the inherent differences between CNNs and ViTs. Although having a quadratic cost, the self-attention mechanism is highly compatible with parallel processing hardware like GPUs and TPUs, processing images globally in a single pass compared to the more local and sequential processing of CNNs. Additionally, ViTs have more regular memory access patterns, potentially reducing overheads for spatial invariance and benefiting from adaptive computation. While these advantages can lead to shorter processing times for ViTs, it is important to notice that the type and number of CPUs and GPUs, as well as their memory size, can greatly impact the efficiency of both ViTs and CNNs.

4.4. Attention Maps

To better understand the previously observed performance variation for different ViT pre-trainings, we compute the attention scores for ViT-B/16 using either supervised IN-21k or self-supervised (DINO) IN-1k pre-training. The results are shown in Figure 5 for three texture images. The scores are obtained as the average of the output of the softmax operation of a transformer block (layer) in the architecture (see Equation (1)), given an input image. In this sense, for the last layer l we obtain the attention matrix $A_l \in \mathbb{R}^{(n+1) \times (n+1) \times s}$ as the output of the softmax of the self-attention mechanism, where s is the number of attention heads. This matrix is then averaged over all attention heads:

$$A_\mu(a, b) = \frac{1}{s} \sum_{z=1}^s A_l(a, b, z). \quad (4)$$

From $A_\mu \in \mathbb{R}^{(n+1) \times (n+1)}$ we first obtain the scores corresponding to the *class token* (first row excluding its first element) and then reshape it according to the number of patches (n), resulting in a $\sqrt{n} \times \sqrt{n}$ matrix. This matrix represents the average attention scores for each token (patch) used on the transformer input. It is then scaled up according to the input image dimensions $w \times h$ (this is the original resolution, not the 224×224 transformation) so that the patches match the original image area. These scores are then used as a mask over the input image for a qualitative analysis of the self-attention operation.

As shown in Figure 5, we selected three different images with similar aspects to better understand the attention mechanism of the different ViTs on textures. The first image contains a wooden statue (labeled as wood), while having a cloudy background. The second image is composed of glass objects (the desired texture), while also having a background with wood texture. The third image does not contain a background and represents only the target texture (wood), which we consider homogeneous here because it lacks background elements or overlapping objects, even though it shows natural grain and pattern variations. While both models can focus on the wooden statue in the first image (with only small differences), the situation is different for the others. The DINO model can effectively focus on the glass texture in the second image but focuses on the wood defects of the last image, which may harm the characterization. On the other hand, ViT with IN-21k pre-training was not able to focus on the glass texture and the attention seems to collapse on the wooden background, but it exhibits a more coherent attention map for the homogeneous wood texture in the last image. This behavior may explain the differences observed between these models in some texture recognition tasks. For instance, IN-21k pre-training achieves a better performance in some cases with homogeneous textures (Outex10 and 12 datasets), while DINO IN-1k pre-training achieves a better performance on the tasks

containing textures in the wild, i.e., with background, multiple objects, etc. (DTD, FMD, and KTH-2-b).

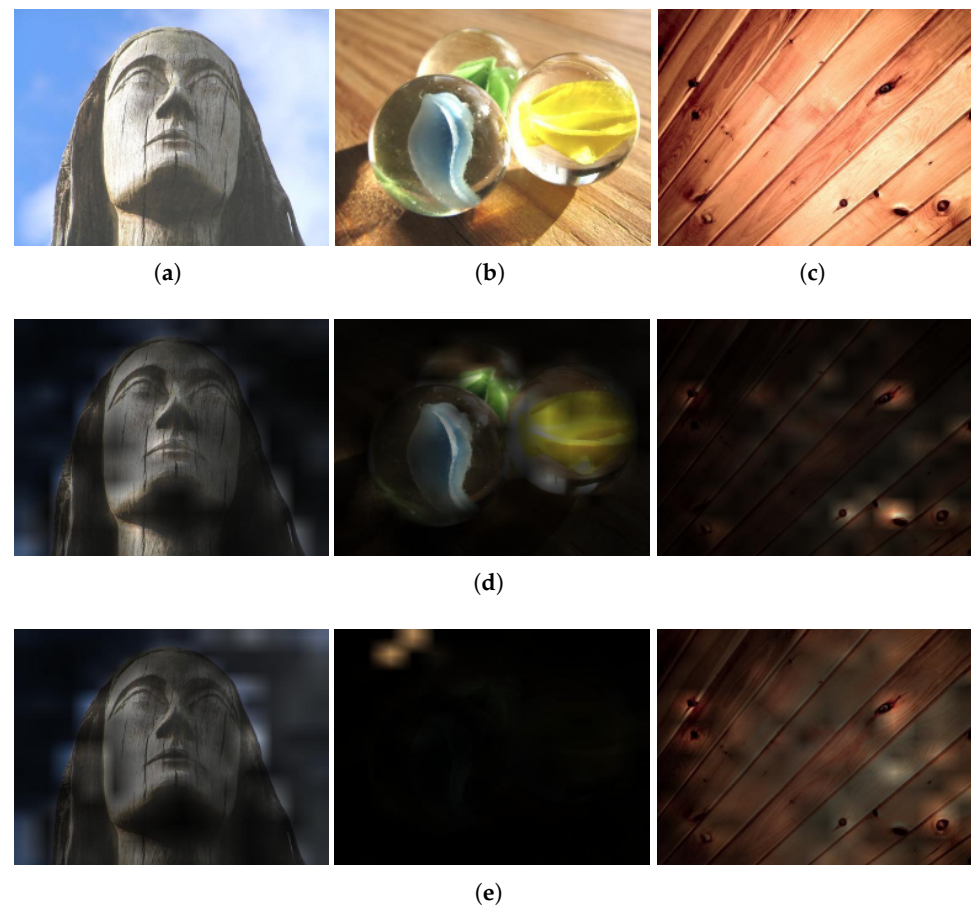


Figure 5. Visualization of attention maps (at the last layer) of different ViT models (**d,e**) for texture samples (**a–c**) from the FMD dataset. (**a**) Wood texture with cloudy background. (**b**) Glass texture with wooden background. (**c**) Homogeneous wood texture. (**d**) ViT-B/16-DINO attention. (**e**) ViT-B/16 (IN-21k) attention.

5. Discussion

The comparative analysis conducted in this study highlights how different ViT architectures respond to the challenges posed by texture datasets, offering insights beyond raw performance metrics. One of the most consistent findings is that model pre-training plays a crucial role in downstream texture classification. ViTs pre-trained on large and diverse datasets (e.g., IN-21k) or through self-supervised methods (e.g., DINO and BeiT) exhibit markedly superior generalization, even when features are extracted without fine-tuning.

Interestingly, the observed performance gains on complex datasets like DTD and FMD suggest that ViTs are particularly adept at modeling high-level visual semantics and capturing global dependencies, key advantages in natural textures with large intra-class variability and cluttered backgrounds. However, their performance is not universally superior: in rotation- and illumination-sensitive datasets such as Outex10, classical methods and well-established CNNs still show strong results. This reinforces the idea that ViTs, while powerful, lack the local spatial inductive biases needed for certain low-level texture properties unless structurally compensated (e.g., via hierarchical or hybrid architectures like Swin).

Another important observation relates to model scalability. Larger ViTs (e.g., ViT-B/16, Swin-B, and BeiTv2-B/16) tend to dominate across datasets, but efficient models like

EfficientFormer-L3 offer a favorable trade-off between inference speed, memory footprint, and accuracy, especially in resource-constrained scenarios. However, smaller mobile-oriented models (e.g., MobileViT and DeiT-Ti) show limited capacity for discriminating fine-grained texture patterns, especially when pre-training is limited.

In practical terms, the results underline that no single model is universally optimal. Task-specific factors, such as dataset characteristics, hardware constraints, and the presence of noise or transformations, should guide model selection. These findings also support further exploration of hybrid strategies that combine ViTs with CNN-like inductive biases or handcrafted descriptors, especially for scenarios where robustness to physical variation is critical.

Finally, the diversity of architectural variants benchmarked in this study highlights the evolving landscape of ViTs, revealing that future improvements might be based not only on scale and data but also on architectural innovation that bridges semantic reasoning with spatial sensitivity, which are both essential in texture understanding.

6. Conclusions

In this work, we explored several aspects of pre-trained ViTs, also known as foundation models, when employed directly for texture analysis by using their class embeddings as image representations. Our analysis shows that ViTs, with their unique architecture and self-attention mechanisms, may provide significant improvements over traditional CNNs and hand-engineered methods in texture recognition tasks. Therefore, the results shed light on the paradigm shift in feature extraction methods in CV. Our experiments compare the features extracted with a variety of ViTs (25 models) for capturing complex texture patterns, their robustness to variations in rotation, scale, and illumination, and the differences between textures filling the whole image or textures in the wild (multiple objects, background, etc.).

We evaluated the ViT models on eight texture recognition tasks, measured their efficiency, analyzed attention maps, and tested three different linear classifiers as their classification heads. ViTs, through their self-attention mechanism, offer a more global perspective of the image data, unlike the local view provided by CNNs, which is an important aspect for texture analysis. We observe that patch embedding and self-supervised learning are important elements to achieve performant texture discrimination. For instance, BeiTv2 and ViT-B/16-DINO demonstrate remarkable performance in general, outperforming other methods, such as ResNet50, with a considerable margin for some tasks. Our results highlight that these models and other ViTs variants can outperform the hand-engineered baselines, ResNet50, and recent methods under IN-1k or IN-21k pre-training regimes. These results corroborate the paradigm shift from CNNs to ViTs seen recently in other CV areas. However, the computation cost of some ViTs may still be a drawback. Some mobile ViT variants may strike a balance between cost and performance, such as the EfficientFormer, as shown by our efficiency analysis. On the other hand, we also show that the inference throughput (i.e., images processed per second) of larger ViTs, in practice, can be superior to ResNets on GPUs, which may be related to transformer mechanisms that are more parallelizable and/or better code optimization.

Although showing promising results, our analyses also indicate a need for new techniques and evaluation of more aspects of transformers regarding textures. Exploring the impacts of different embedding sizes, image resolutions, and model depths will help consolidate their utility in texture analysis. Another aspect is the need for optimized ViT models that balance performance with computational efficiency, making them more accessible for real-world applications. Nevertheless, ViTs emerge as powerful candidates in texture analysis, offering new perspectives and capabilities and corroborating their groundbreaking

results in other CV areas. New feature aggregation techniques specifically designed for ViTs and texture may significantly improve the SOTA of texture analysis. Furthermore, as ViTs continue to evolve, they hold the promise of impacting various industries and tasks that rely on texture recognition models.

As a future research direction, understanding how deep models such as ViTs capture and represent texture-specific patterns across different layers could provide valuable insights into their decision-making processes. Another key direction involves the expansion and enrichment of texture datasets. Most existing datasets are either limited in scale, lack diversity in texture types and contexts, or are constrained to controlled environments. Developing new large-scale, in-the-wild texture datasets with richer annotations (e.g., semantic labels, hierarchical categories, or physical properties) would allow for more robust training and enhance the generalization capability of deep models. Advancing in these directions will help bridge the gap between academic benchmarks and practical deployment in real-world texture-related applications.

Author Contributions: Conceptualization, L.S., A.S., K.M.Z., L.C.R., B.D.B. and O.M.B.; methodology, L.S., A.S., K.M.Z. and L.C.R.; validation, L.S., A.S. and K.M.Z.; formal analysis, L.S., K.M.Z. and L.C.R.; investigation, L.S., A.S., K.M.Z. and L.C.R.; resources, L.S., L.C.R., B.D.B. and O.M.B.; writing—original draft preparation, L.S., A.S., K.M.Z. and L.C.R.; writing—review and editing, L.S., L.C.R., B.D.B. and O.M.B.; supervision, L.C.R., B.D.B. and O.M.B.; funding acquisition, L.C.R., B.D.B. and O.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: L.S. acknowledges funding from FAPESP (grants #2023/10442-2 and #2024/00530-4). L.C.R. acknowledges support from FAPESP (grant #2023/04583-2). K.M.Z. acknowledges support from CAPES (grant #88887.631085/2021-00) and FAPESP (grant #2022/03668-1). B.D.B. received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” program. O.M.B. acknowledges support from CNPq (Grant #305610/2022-8) and FAPESP (grants #2018/22214-6 and #2021/08325-2).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Scabini, L.F.; Ribas, L.C.; Bruno, O.M. Spatio-spectral networks for color-texture analysis. *Inf. Sci.* **2020**, *515*, 64–79. [\[CrossRef\]](#)
2. Liu, L.; Chen, J.; Fieguth, P.; Zhao, G.; Chellappa, R.; Pietikäinen, M. From BoW to CNN: Two decades of texture representation for texture classification. *Int. J. Comput. Vis.* **2019**, *127*, 74–109. [\[CrossRef\]](#)
3. Humeau-Heurtier, A. Texture feature extraction methods: A survey. *IEEE Access* **2019**, *7*, 8975–9000. [\[CrossRef\]](#)
4. Pietikäinen, M.; Ojala, T. Texture analysis in industrial applications. In *Image Technology: Advances in Image Processing, Multimedia and Machine Vision*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 337–359.
5. Kassner, A.; Thornhill, R. Texture analysis: A review of neurologic MR imaging applications. *Am. J. Neuroradiol.* **2010**, *31*, 809–816. [\[CrossRef\]](#)
6. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [\[CrossRef\]](#)
7. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [\[CrossRef\]](#)
8. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [\[CrossRef\]](#)
9. Scabini, L.F.; Condori, R.H.; Gonçalves, W.N.; Bruno, O.M. Multilayer complex network descriptors for color-texture characterization. *Inf. Sci.* **2019**, *491*, 30–47. [\[CrossRef\]](#)

10. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
11. Touvron, H.; Cord, M.; Jégou, H. DeiT iii: Revenge of the vit. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022, Proceedings, Part XXIV; Springer: Cham, Switzerland, 2022; pp. 516–533.
12. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
13. Bao, H.; Dong, L.; Piao, S.; Wei, F. BEiT: BERT Pre-Training of Image Transformers. In Proceedings of the International Conference on Learning Representations, Online, 25–29 April 2022.
14. Peng, Z.; Dong, L.; Bao, H.; Ye, Q.; Wei, F. BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers. *arXiv* **2022**, arXiv:2208.06366.
15. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 26183–26197.
16. Hong, W.; Lao, J.; Ren, W.; Wang, J.; Chen, J.; Chu, W. Training Object Detectors From Scratch: An Empirical Study in the Era of Vision Transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 4662–4671.
17. Wang, H.; Xie, S.; Lin, L.; Iwamoto, Y.; Han, X.H.; Chen, Y.W.; Tong, R. Mixed Transformer U-Net for Medical Image Segmentation. In Proceedings of the ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 2390–2394.
18. Lu, Z.; Li, J.; Liu, H.; Huang, C.; Zhang, L.; Zeng, T. Transformer for Single Image Super-Resolution. *arXiv* **2021**, arXiv:2206.02680.
19. Ojala, T.; Pietikainen, M.; Maenpää, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 971–987. [[CrossRef](#)]
20. Hoang, M.A.; Geusebroek, J.M.; Smeulders, A.W. Color texture measurement and segmentation. *Signal Process.* **2005**, *85*, 265–275. [[CrossRef](#)]
21. Backes, A.R.; Casanova, D.; Bruno, O.M. Color texture analysis based on fractal descriptors. *Pattern Recognit.* **2012**, *45*, 1984–1992. [[CrossRef](#)]
22. Zhang, J.; Tan, T. Brief review of invariant texture analysis methods. *Pattern Recognit.* **2002**, *35*, 735–747. [[CrossRef](#)]
23. Cantero, S.V.A.B.; Gonçalves, D.N.; dos Santos Scabini, L.F.; Gonçalves, W.N. Importance of vertices in complex networks applied to texture analysis. *IEEE Trans. Cybern.* **2018**, *50*, 777–786. [[CrossRef](#)]
24. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In Proceedings of the International Conference on Machine Learning, PMLR, Beijing, China, 22–24 June 2014; pp. 647–655.
25. Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; Vedaldi, A. Describing textures in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3606–3613.
26. Scabini, L.; Zielinski, K.M.; Ribas, L.C.; Goncalves, W.N.; De Baets, B.; Bruno, O.M. RADAM: Texture Recognition through Randomized Aggregated Encoding of Deep Activation Maps. *Pattern Recognit.* **2023**, *143*, 109802. [[CrossRef](#)]
27. Zhang, H.; Xue, J.; Dana, K. Deep TEN: Texture encoding network. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2896–2905.
28. Yang, Z.; Lai, S.; Hong, X.; Shi, Y.; Cheng, Y.; Qing, C. DFAEN: Double-order knowledge fusion and attentional encoding network for texture recognition. *Expert Syst. Appl.* **2022**, *209*, 118223. [[CrossRef](#)]
29. Raghu, M.; Unterthiner, T.; Kornblith, S.; Zhang, C.; Dosovitskiy, A. Do vision transformers see like convolutional neural networks? *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12116–12128.
30. Bai, Y.; Mei, J.; Yuille, A.L.; Xie, C. Are transformers more robust than cnns? *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26831–26843.
31. Yang, F.; Yang, H.; Fu, J.; Lu, H.; Guo, B. Learning texture transformer network for image super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5791–5800.
32. Yao, C.; Zhang, S.; Yang, M.; Liu, M.; Qi, J. Depth super-resolution by texture-depth transformer. In Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 5–9 July 2021; pp. 1–6.
33. Yang, G.; Qian, Y.; Liu, H.; Tang, B.; Qi, R.; Lu, Y.; Geng, J. MSFusion: Multistage for remote sensing image spatiotemporal fusion based on texture transformer and convolutional neural network. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 4653–4666. [[CrossRef](#)]
34. Zhang, X.; Saniie, J. Material Texture Recognition using Ultrasonic Images with Transformer Neural Networks. In Proceedings of the 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 14–15 May 2021; pp. 1–5.
35. Soleymani, M.; Bonyani, M.; Mahami, H.; Nasirzadeh, F. Construction material classification on imbalanced datasets using Vision Transformer (ViT) architecture. *arXiv* **2021**, arXiv:2108.09527.

36. Tao, X.; Adak, C.; Chun, P.J.; Yan, S.; Liu, H. ViTALnet: Anomaly on Industrial Textured Surfaces With Hybrid Transformer. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–13. [CrossRef]
37. Xu, W.; Xu, Y.; Chang, T.; Tu, Z. Co-scale conv-attentional image transformers. In Proceedings of the the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 9981–9990.
38. Mehta, S.; Rastegari, M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
39. Mehta, S.; Rastegari, M. Separable Self-attention for Mobile Vision Transformers. *arXiv* **2022**, arXiv:2206.02680. [CrossRef]
40. Li, Y.; Yuan, G.; Wen, Y.; Hu, E.; Evangelidis, G.; Tulyakov, S.; Wang, Y.; Ren, J. EfficientFormer: Vision Transformers at MobileNet Speed. *arXiv* **2022**, arXiv:2206.01191.
41. Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 9650–9660.
42. Chen, X.; Hsieh, C.J.; Gong, B. When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations. In Proceedings of the International Conference on Learning Representations, Online, 25–29 April 2022.
43. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 10347–10357.
44. Chen, C.F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 357–366.
45. d’Ascoli, S.; Touvron, H.; Leavitt, M.L.; Morcos, A.S.; Biroli, G.; Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 2286–2296.
46. Hatamizadeh, A.; Yin, H.; Kautz, J.; Molchanov, P. Global context vision transformers. *arXiv* **2022**, arXiv:2206.09959.
47. Li, Y.; Wu, C.Y.; Fan, H.; Mangalam, K.; Xiong, B.; Malik, J.; Feichtenhofer, C. MViTv2: Improved Multiscale Vision Transformers for Classification and Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4804–4814.
48. Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; Jégou, H. Going deeper with image transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 32–42.
49. Ali, A.; Touvron, H.; Caron, M.; Bojanowski, P.; Douze, M.; Joulin, A.; Laptev, I.; Neverova, N.; Synnaeve, G.; Verbeek, J.; et al. Xcit: Cross-covariance image transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 20014–20027.
50. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 10012–10022.
51. Li, J.; Xia, X.; Li, W.; Li, H.; Wang, X.; Xiao, X.; Wang, R.; Zheng, M.; Pan, X. Next-vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios. *arXiv* **2022**, arXiv:2207.05501.
52. Yun, S.; Ro, Y. Shvit: Single-head vision transformer with memory efficient macro design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 5756–5767.
53. Tschannen, M.; Gritsenko, A.; Wang, X.; Naeem, M.F.; Alabdulmohsin, I.; Parthasarathy, N.; Evans, T.; Beyer, L.; Xia, Y.; Mustafa, B.; et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv* **2025**, arXiv:2502.14786.
54. Alabdulmohsin, I.; Zhai, X.; Kolesnikov, A.; Beyer, L. Getting vit in shape: Scaling laws for compute-optimal model design, 2024. *arXiv* **2023**, arXiv:2305.13035.
55. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
56. Fix, E.; Hodges, J.L. *Discriminatory Analysis, Nonparametric Estimation: Consistency Properties*; Report 4, Project n° 21–49; USAF School of Aviation Medicine: Randolph Field, TX, USA, 1951; Volume 4.
57. Ripley, B.D. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, UK, 2007.
58. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
59. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems 32, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
60. Wightman, R. PyTorch Image Models. 2019. Available online: <https://github.com/rwightman/pytorch-image-models> (accessed on 10 November 2024).

61. Perronnin, F.; Sánchez, J.; Mensink, T. Improving the fisher kernel for large-scale image classification. In Proceedings of the Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, 5–11 September 2010, Proceedings, Part IV 11; Springer: Berlin/Heidelberg, Germany, 2010; pp. 143–156.
62. Badanidiyoor, A.R.; Naravi, G.K. θ (1) time complexity parallel local binary pattern feature extractor on a graphical processing unit. *ICIC Express Lett.* **2019**, *13*, 867–874.
63. 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, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
64. Beyer, L.; Zhai, X.; Royer, A.; Markeeva, L.; Anil, R.; Kolesnikov, A. Knowledge distillation: A good teacher is patient and consistent. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 10925–10934.
65. Huovinen, S.; Pietikäinen, M.; Ojala, T.; Kyllönen, J.; Viertola, J.; Mäenpää, T. Outex–New Framework for Empirical Evaluation of Texture Analysis Algorithms. In Proceedings of the 16th International Conference on Pattern Recognition, Quebec, QC, Canada, 11–15 August 2002; Volume 1, pp. 701–706.
66. Sharan, L.; Rosenholtz, R.; Adelson, E. Material perception: What can you see in a brief glance? *J. Vis.* **2010**, *9*, 784–784. [[CrossRef](#)]
67. Caputo, B.; Hayman, E.; Mallikarjuna, P. Class-specific material categorisation. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China, 17–21 October 2005; Volume 2, pp. 1597–1604.
68. Lyra, L.O.; Fabris, A.E.; Florindo, J.B. A multilevel pooling scheme in convolutional neural networks for texture image recognition. *Appl. Soft Comput.* **2024**, *152*, 111282. [[CrossRef](#)]
69. Chen, Z.; Quan, Y.; Xu, R.; Jin, L.; Xu, Y. Enhancing texture representation with deep tracing pattern encoding. *Pattern Recognit.* **2024**, *146*, 109959. [[CrossRef](#)]
70. Florindo, J.B. Fractal pooling: A new strategy for texture recognition using convolutional neural networks. *Expert Syst. Appl.* **2024**, *243*, 122978. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.