I.J. Information Technology and Computer Science, **2026, 2, 1-18**
Published Online on April 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2026.02.01

A Study of Using Convolutional Neural Networks in Fractal Dimension Estimation of Grayscale and Color Images

Moheb R. Girgis*
Department of Computer Science, Faculty of Science, Minia University, El-Minia, 61519, Egypt
E-mail: moheb.girgis@mu.edu.eg
ORCID iD: http://orcid.org/0000-0002-2159-6881
*Corresponding author

Al Hussien Seddik Saad
Department of Computer Science, Faculty of Science, Minia University, El-Minia, 61519, Egypt
E-mail: al.hussien_seddik@mu.edu.eg

Mohammed M. Talaat
Department of Computer Science, Faculty of Science, Minia University, El-Minia, 61519, Egypt
E-mail: Talaat_eng@yahoo.com

Received: 07 August 2025; Revised: 19 November 2025; Accepted: 17 January 2026; Published: 08 April 2026

Abstract: Fractal dimension (FD) estimation is widely used to characterize image complexity and self-similarity in image analysis and texture characterization. Traditional FD estimators such as Box-Counting (BC) and Differential Box-Counting (DBC) are simple and efficient but can be sensitive to scale selection, resolution, and noise. This paper investigates the effectiveness of using a convolutional neural network (CNN) in FD estimation compared to traditional methods. To this end, we have developed a CNN-based method for FD estimation under a fair and reproducible evaluation design. First, we have included an analytic-fractals benchmark (such as Sierpinski and Koch families) with closed-form FD values for independent evaluation. Second, for large-scale Julia/Mandelbrot images, FD labels are treated as reference estimates computed from multiple BC/DBC parameter settings and reported as mean ± standard deviation to quantify label uncertainty. We additionally assess behavior on an external natural-texture dataset and evaluate robustness under controlled degradations (noise, blur, compression, and downsampling). Performance is reported on large test sets using MAE/RMSE with 95% confidence intervals (bootstrap), together with per-image inference time under clearly specified hardware settings. Results indicate that the proposed CNN-based method provides stable FD estimation and fast inference, particularly under noise and resolution variations.

Index Terms: Fractal Dimension, Box-Counting, Differential Box-Counting, Deep Learning, Convolutional Neural Networks.

1. Introduction

Fractal dimension (FD) is a mathematical measure of irregularity that has been used to quantify the self-similar structures of complex and chaotic systems, both natural and artificially created. While dimensions in Euclidean space are generally integers, fractal dimensions can be non-integers describing the complexity of an object that appears self-similar at different levels of magnification. This trait makes FD valuable in various areas including medical imaging, texture analysis, and environmental science, since it can characterize complex structures that cannot be characterized with conventional geometric measures [1-3]. By estimating the FD of an image or structure, researchers can obtain a quantitative measure of its detail or roughness, and individuality [4].

The Box-Counting (BC) method [1] is one of the most popular methods for estimating FDs, due to its simplicity and the ability to measure a variety of structures. However, it can be imprecise for highly complex patterns, especially if image resolution or noise influences box occupancy [5,6].

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

The Differential Box-Counting (DBC) method was introduced by Sarkar and Chaudhuri [5] to tackle some of the limitations associated with BC. It is appropriate for estimating the FD in grayscale or textured images.

These traditional methods have been successfully applied to estimate FDs; however, the recent breakthroughs in deep learning have provided new strategies to tackle this task, such as Convolutional Neural Networks (CNNs), which are data-driven techniques. CNNs are a type of deep learning model which automatically detects patterns in visual data using a series of filters applied to an input image, while also localizing both local and global features [7].

Although FD is an asymptotic, scale-invariant quantity, CNNs naturally learn hierarchical multi-scale representations via stacked convolutions and pooling; the effective receptive field grows with depth, enabling aggregation of structures across multiple resolutions. This multi-scale representation is consistent with the scale-space behavior used in classical FD estimation, which motivates using CNNs as a surrogate estimator of FD-related complexity.

The aim of this paper is to study CNN-based FD estimation under a fair and reproducible evaluation design. In addition to large-scale Julia/Mandelbrot images, we evaluate the proposed model on analytic fractals with known closed-form FD values to avoid circular comparisons. For synthetic fractal images where closed-form FD is not available, labels are treated as reference estimates computed from multiple classical estimators and parameter settings, and their uncertainty is reported.

Main contributions of this work are: (1) a CNN-based FD estimator motivated by multi-scale feature learning; (2) a mixed evaluation framework including analytic-fractal benchmarks with closed-form FD; (3) uncertainty-aware reference labeling for Julia/Mandelbrot images using multiple BC/DBC settings (mean ± std); (4) robustness evaluation under controlled degradations; and (5) fair runtime reporting separating training cost from per-image inference.

The rest of the paper is organized as follows: Section 2 presents a review of the related work in FD estimation; Section 3 gives a background on the traditional FD estimation methods, BC and DBC; Section 4 presents the implementation of these traditional FD estimation methods for grayscale and RGB images; Section 5 presents the implementation details of the proposed CNN-based approach for FD estimation of images; Section 6 presents the results of the experiments that we have conducted to measure the prediction accuracy of the proposed CNN-based FD estimation method, and evaluate its performance compared to the BC and DBC methods, for grayscale and RGB images; Finally, Section 7 provides the conclusion of the presented work.

2. Related Works

Fractal dimension estimation has been extensively studied for analyzing self-similar structures in medical imaging, texture classification, and pattern recognition. Traditional estimators such as BC and DBC are widely used due to their simplicity, yet their estimates can be sensitive to scale selection, resolution, and preprocessing (e.g., thresholding) and may degrade under noise. These factors motivate uncertainty-aware labeling and robustness evaluation when FD is used for training or benchmarking learning-based models.

Sarkar and Chaudhuri [5] introduced the Differential Box-Counting (DBC) method, which improves on the traditional BC method by considering the intensity variations within different sub-regions of an image, making it more suitable for grayscale and textured images.

Other studies [4,6] have highlighted the limitations of traditional methods in handling noisy and high-resolution images, motivating the need for more robust techniques.

Liu et al. [8] presented an enhancement to the DBC method and compared it against other DBC-based algorithms using texture images. Li [9] extended the DBC approach to five-dimensional Euclidean hyperspace, to estimate fractal dimension for color texture images.

For color images, FD can be estimated on a luminance/grayscale representation or by computing channel-wise FD values and aggregating them (e.g., mean/median) to obtain a single descriptor for color textures; this formulation is commonly used in color FD estimation studies such as [9] and ANN-based color-FD approaches [10].

While traditional fractal dimension estimation methods, such as BC and DBC, are widely used for analyzing complex textures, recent research has explored the use of deep learning to enhance or complement fractal-based analysis. Deep learning models are typically employed to learn features that are sensitive to fractal structures, capturing scale-invariant or self-similar patterns in image data.

Xu et al. [11] introduced a Fractal Dimension Invariant Filtering (FDIF) approach designed to preserve local fractal structures in images during the feature extraction process. This method has been integrated with convolutional layers to improve performance in tasks like curve detection and texture classification, demonstrating the potential of fractal-aware deep learning.

Valle et al. [12] applied deep neural networks to approximate fractal basins of attraction in chaotic systems. Although not estimating the fractal dimension numerically, their model learned to represent fractal geometry patterns efficiently, offering a path toward hybrid systems that combine fractal theory with deep learning for complex dynamics modeling.

Wang et al. [13] have proposed a fractal reconstruction method for the surface image of ultra-high temperature ceramics after repeated thermal shocks. A fractal CNN model based on deep learning was established to realize automatic recognition of the classification of thermal shock cycles of ultra-high temperature ceramics.

Zhou et al. [14] proposed an ANN model for estimating fractal dimensions of machined surfaces, achieving high accuracy with minimal computation time. Padhy et al. [10] have proposed an ANN-based methodology to compute fractal

dimension of color images using a probabilistic approach without losing their color information, and they have used TensorFlow for classification of images in this application.

In addition to deep CNN models, prior work has also used simpler regressors such as shallow ANNs for FD-related estimation tasks (e.g., [10,14]). Therefore, comparing CNNs against simpler baselines is important to justify model complexity.

3. Traditional Fractal Dimension Estimation Methods

This section gives a background on the traditional FD estimation methods, BC and DBC.

3.1. Box-Counting Method

The Box-Counting (BC) Method is a quantitative method for estimating the fractal dimension, particularly in cases where an irregular shape or complex image (like histology slides) cannot be accurately modeled with a simple Euclidean dimension. Specifically, it is a useful technique for scaling characteristics of self-similar structures and patterns in medical and biological imaging.

The BC method includes the following main steps:

- Grid Overlay: It imposes a grid of boxes of size ϵ over the structure/image. Only those boxes that contain part of the structure are counted for each box size.
- Scale Variation: This procedure is repeated over a set of box sizes ϵ, so that details at different scales are preserved [1].
- Fractal Dimension Calculation: Based on the relationship between box counts N(ϵ) and ϵ, the fractal dimension, FD, can be estimated by using (1).

$$FD = \lim_{\varepsilon \to 0} \frac{\log\left(N(\varepsilon)\right)}{\log\left(1/\varepsilon\right)} \tag{1}$$

Such an approach aids in quantifying the structural complexity of patterns found within images and has been particularly useful for analyzing medical imaging patterns, with the BC method offering information regarding biological packing and pathology [1].

3.2. Differential Box-Counting

The Differential Box-Counting (DBC) Method is a modification of the classical BC method aimed at obtaining a more precise value of the fractal dimension, which works especially well for grayscale images since the system complexity is not determined only by shapes and structures. Sarkar and Chaudhuri [5] proposed this method to overcome some of the deficiencies of the standard BC method by taking into account not only spatial distribution of grey level intensities but also their variation values within each grid cell.

The steps of the DBC method are described below:

- 3D Grid Construction: For a grayscale image, the DBC method considers the image as a three-dimensional surface, where the x and y axes represent spatial dimensions, and the z-axis represents intensity levels. The image is divided into a grid of M×M cells, each of which corresponds to a section of the image with a defined range of intensity values.
- Intensity Range and Differential Count: The minimum and maximum intensity levels, I_{min} and I_{max} , are determined for each grid cell. Then, for every cell (i, j), the count $n(i,j)$ is computed as follows:

$$n(i,j) = \left\lfloor \frac{I_{\max} - I_{\min}}{\Delta I} \right\rfloor + 1 \tag{2}$$

where ΔI represents the intensity interval defined for each box.

- Fractal Dimension Estimation: The relationship between the number of filled boxes N(M) and scale M is constructed by repeatedly executing the differential box count for a diversity of grid sizes. Then, FD can be estimated from the least-squares linear fit of log (N(M)) against log (1 / M), by using (3).

$$FD = \lim_{M \to \infty} \frac{\log\left(N(M)\right)}{\log\left(1/M\right)} \tag{3}$$

In this manner, the DBC method is able to characterize variations in intensity and spatial distribution, which makes it even more applicable in medical images containing critical texture and/or intensity differences (e.g. histological image analysis, radiology image analysis) [15].

4. Implementation of Traditional FD Estimation Methods for Grayscale/RGB Images

In this section, we present the implementation of the traditional FD estimation methods, BC and DBC, for grayscale and RGB images.

4.1. FD Estimation of Grayscale Images

A. FD Estimation of Grayscale Images Using Box-Counting

The steps of the algorithm for FD estimation of grayscale images using the BC method are shown in Fig. 1.

Algorithm 1 Grayscale Box-Counting

Begin
1. Read a grayscale image
   1.1 Binarize the image using Otsu's global thresholding method to obtain a binary mask.
2. Display image
3. Perform Box-Counting Analysis.
   3.1 Execute the boxcount() function on the image data c with the option 'slope'
   3.2 Obtain the number of boxes n and their corresponding sizes r required to cover the foreground pixels of the image at different scales.
4. Compute the FD from the box-counting results. The FD represents the scaling behavior of the image and indicates its self-similarity and complexity.
5. Display the resulting FD.

End.

Fig.1. Grayscale Box-counting algorithm

B. FD Estimation of Grayscale Images Using Differential Box-Counting

The steps of the algorithm for FD estimation of grayscale images using DBC are shown in Fig. 2.

Algorithm 2 Grayscale DBC

Begin
1. Read a grayscale image
2. Apply the Differential Box-Counting method at multiple scales:
   For each scale (different box sizes), count the number of boxes (n) that are required to cover the non-zero pixels of the image
3. Calculate the FD:
   3.1 Use the log-log plot of box sizes (r) versus box counts (n).
   3.2 Compute the slope of the plot using the differential approach to estimate the fractal dimension.
4. Display the resulting FD.

End.

Fig.2. Grayscale DBC algorithm

4.2. FD Estimation of RGB Images

Although the algorithms presented above are for grayscale images, RGB images can be handled by operating on each color channel independently. For an RGB image, we compute the FD for each channel (FD_R, FD_G, FD_B) using the selected estimator (BC or DBC), and then aggregate them into a single descriptor: $FD_{RGB} = (FD_R + FD_G + FD_B)/3$. This aggregation provides one FD value per color image and reduces channel-specific variability. All RGB images are resized to a fixed resolution (e.g., 256×256) and normalized to [0,1] prior to estimation. For BC, channels are binarized using a reported thresholding rule (Otsu's method) to obtain a foreground mask, whereas DBC is computed directly from normalized intensities without binarization.

In this section, we present the implementation of the traditional FD estimation methods, BC and DBC, for RGB images.

A. FD Estimation of RGB Images Using Box-Counting

The steps of the algorithm for FD estimation of RGB images using BC are shown in Fig. 3.

B. FD Estimation of RGB Images Using Differential Box-Counting

The steps of the algorithm for FD estimation of RGB images using DBC are shown in Fig. 4.

5. The Proposed CNN-Based Approach for FD Estimation

CNNs can capture and learn high-level abstractions in images, and can extract self-similar features, an important property of fractals, by applying multiple convolutional filters. They can also be effectively trained to evaluate texture patterns and variations indicating fractal structures. This section presents the implementation details of the proposed CNN-based approach for FD estimation of images.

Algorithm 3 RGB Box-Counting

Begin

1. Read an RGB image.
   The image is read as an RGB image (which has 3 channels: Red, Green, Blue).
2. Display the Image
   Show the image to the user to visually inspect it.
3. Resize the image to 256×256 and normalize to [0,1]
4. Perform Box-Counting Analysis
   4.1 For each of the 3 channels (Red, Green, Blue) Do:
   4.1.1 Convert the channel to a binary image using Otsu's global thresholding method to obtain a foreground mask.
   4.1.2 Use the box-counting method (the boxcount() function) to calculate the number of boxes (n) and the box sizes (r).
5. Calculate FD
   5.1 Using the results from the box-counting analysis (n and r), compute the fractal dimension for each channel.
   5.2 This gives the FDs for the red, green, and blue channels individually.
6. Combine Results for RGB:
   The overall fractal dimension of the RGB image can be obtained by averaging the fractal dimensions of the three channels (Red, Green, and Blue), $FD_{RGB} = (FD_R + FD_G + FD_B)/3$.
7. Display the resulting FD.

End.

Fig.3. RGB Box-counting algorithm

Algorithm 4 RGB DBC

Begin

1. Read an RGB image
2. Extract RGB Channels:
   Separate the image into its individual color channels: Red, Green, and Blue. These will be treated as grayscale images for the differential box-counting process.
3. Apply the Differential Box-Counting method at multiple scales:
   3.1 For each channel (R, G, B), apply the differential box-counting algorithm by covering the non-zero elements of the image with boxes of varying sizes.
   3.2 For each box size, count how many boxes are required to cover the non-zero elements.
   3.3 This step is repeated for different box sizes.
4. Calculate the Fractal Dimension:
   4.1 For each scale (box size), calculate the number of boxes n(r) required to cover the non-zero elements in each of the channels.
   4.2 The box size will vary from large to small, and at each scale, the number of boxes required is counted.
   4.3 Use the log-log plot of box sizes (r) versus box counts (n).
   4.4 Compute the slope of the plot using the differential approach to estimate the fractal dimension.
5. Combine Results for RGB:
   5.1 For each color channel (R, G, B), compute the fractal dimension.
   5.2 The overall fractal dimension of the RGB image can be obtained by averaging the fractal dimensions of the three channels (Red, Green, and Blue), $FD_{RGB} = (FD_R + FD_G + FD_B)/3$.
6. Display the result FD.

End.

Fig.4. RGB DBC algorithm

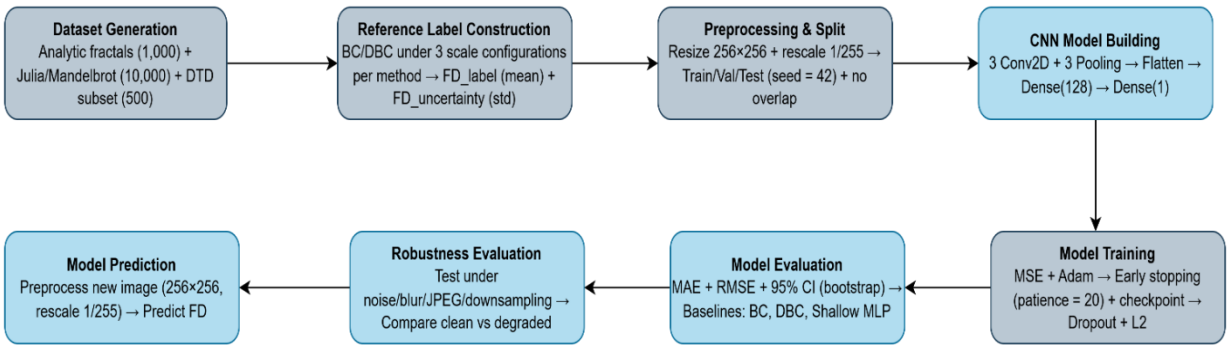


Fig.5. The components of the proposed CNN-based approach

5.1. The Proposed Approach Structure

The main components of the proposed approach, as shown in Fig. 5, are:

- Dataset Generation & Preparation: To support supervised learning and avoid circular evaluation, we employ a mixed evaluation design that combines synthetic fractals with real-texture data. (i) Analytic-fractals dataset: we generate 1,000 analytic-fractal images from three families (Sierpinski triangle, Sierpinski carpet, and Koch curve) with closed-form FD values [1,2,16]. We use 200 analytic-fractals images as training and keep the remaining 800 analytic-fractals images strictly held out as an independent benchmark test set. (ii) Julia/Mandelbrot dataset: we generate 10,000 images by varying fractal parameters and rendering settings. Since closed-form FD values are not available for these instances, each image is assigned a reference FD label computed by aggregating multiple classical FD estimates obtained from BC and DBC under three fixed multi-scale configurations per method; the final label is the mean of all estimates and the uncertainty is reported as the standard deviation (mean ± std) to capture sensitivity to scale selection and image resolution [1,5,6,16]. (iii) External real-texture test set: we use a fixed subset of 500 images sampled from the Describable Textures Dataset (DTD) using a fixed random seed (seed = 42). This subset is used for external testing only to assess generalization to real-world textures [17]. (iv) Controlled robustness evaluation: we create degraded versions of the Julia/Mandelbrot internal test images using fixed corruption types and severity levels (Gaussian noise, Gaussian blur, JPEG compression, and downsampling) and compare performance on clean versus degraded samples [18].
- Reference Label Construction and Uncertainty: For the Julia/Mandelbrot dataset, closed-form FD values are not available for the generated image instances. Therefore, we construct a reference FD label by aggregating multiple classical estimates under controlled multi-scale settings, rather than treating a single BC output as exact ground truth. All images are resized to a fixed resolution of 256×256 before label computation. For BC, we obtain a binary foreground mask using Otsu thresholding applied consistently across all images [19]. We evaluate multiple scales using dyadic box sizes $r = 2^k$ selected within the resized image dimension, while excluding extreme scales that lead to trivial or degenerate counts for log–log fitting. For each selected r, we compute $N(r)$, the number of boxes that intersect the foreground set. Linear fitting is performed only on scales for which the box counts are non-degenerate, to avoid degenerate counts at extreme scales. The BC estimate is computed as the slope of the ordinary least-squares fit in the log–log domain [1,6,16] using (4).

$$\hat{D}_{BC} = \text{slope}\left(\log N(r) \text{ vs. } \log\left(\frac{1}{r}\right)\right) \tag{4}$$

For Differential Box-Counting (DBC), we apply the standard DBC procedure on the grayscale intensity surface [5]. Using dyadic grid sizes $r = 2^k$ within the resized image dimension and excluding extreme scales analogously, we compute a scale-dependent count $N_{\mathrm{DBC}}(r)$ for each selected r, and estimate the DBC FD using (5).

$$\hat{D}_{DBC} = \text{slope}\left(\log N_{DBC}(r) \text{ vs. } \log\frac{1}{r}\right) \tag{5}$$

The same non-degeneracy criterion is applied when selecting scales for the DBC fit. To reduce dependence on a particular scale range and to quantify sensitivity to scale selection, each estimator is evaluated under three fixed multi-scale configurations per method by using three predefined scale ranges derived from the same dyadic scale set, where each configuration excludes one or both extreme scale ranges. Across both estimators and all configurations, we obtain a set of FD estimates for each image using (6).

$$S(I) = \left\{ \hat{D}_{m,c}(I) \,\middle|\, m \in \{\mathrm{BC}, \mathrm{DBC}\}, c \in \{1,2,3\} \right\} \tag{6}$$

We define the reference training label as the mean of these estimates:

$$D_{label}(I) = \frac{1}{|S(I)|} \sum_{\hat{D} \in S(I)} \hat{D} \tag{7}$$

and we report the associated uncertainty as the standard deviation:

$$D_{uncertainty}(I) = \sqrt{\frac{1}{|S(I)|-1} \sum_{\hat{D} \in S(I)} \left(\hat{D} - D_{label}(I)\right)^2} \tag{8}$$

Reporting $D_{\text{label}}(I) \pm D_{\text{uncertainty}}(I)$ provides a quantitative indication of sensitivity to estimator choice and scale-range selection, directly addressing the dependence of BC-style estimators on scale selection and image resolution.

- **CNN Model Building:** A CNN generally consists of multiple convolutional layers followed by pooling layers to capture both localized and global features at different levels. This hierarchical feature extraction supports FD approximation because multi-scale convolution and pooling capture scale-dependent texture and occupancy patterns that correlate with fractal dimension.

The architecture of the utilized CNN model is shown in Fig. 6. It includes the following layers:

- **3 Conv2D Layers:** They represent spatial patterns over various scales. The numbers of filters used in the three Conv2D layers are 32, 64, and 128, respectively, and each layer uses a rectified linear unit (ReLU) activation function.
- **3 Pooling Layers:** These are used to down-sample the feature maps while preserving dominant structural responses relevant to fractal features.
- **Dense layers:** The last layer of the architecture is a fully connected layer projected to a single output node with a linear activation function. These dense layers are used to map the extracted features to the predicted fractal dimension.

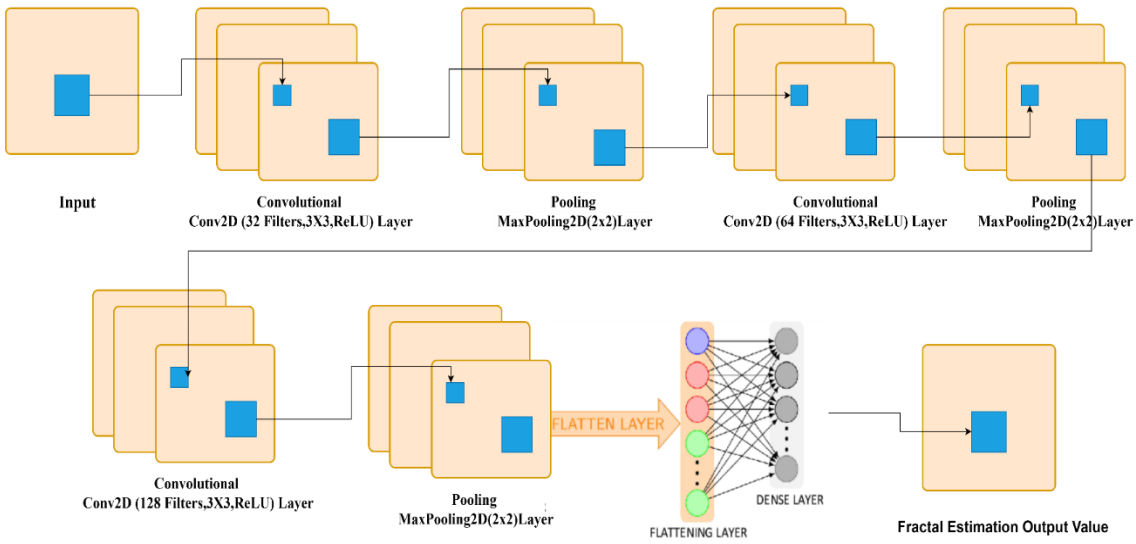


Fig.6. The architecture of the proposed CNN model for FD Estimation

- Model Training: The CNN is trained using supervised learning on labeled images. The loss function, Mean Squared Error (MSE), is used to minimize the difference between the fractal dimensions predicted by the CNN and the reference FD labels. The optimization algorithm Adam is used during training to achieve proper fractal dimension prediction.
- Input images are resized to 256×256 and normalized to [0,1] using rescale = 1/255. Training is run for up to 1000 epochs with early stopping on validation loss (patience = 20) and best-weight checkpointing. Regularization is applied using dropout and L2 weight decay, and training/validation loss curves are reported to assess overfitting.

- Model Evaluation: The performance of the proposed model is evaluated using MAE and RMSE on the held-out test sets. We report 95% confidence intervals for MAE using bootstrap resampling on test predictions. Results are also compared against BC, DBC, and a shallow MLP baseline using the same test sets.

Table 1 summarizes the parameters of the utilized CNN model.

Table 1. Summary of the parameters of the utilized CNN model

Parameter	Value
Model architecture	3 Conv2D layers → 3 Pooling layers → Flatten → Dense (128) → Dense (1)
Activation functions	Rectified Linear Unit (ReLU) for Conv/Dense, Linear for output (regression)
Loss Function	Mean Squared Error (MSE)
Optimizer	Adam
Evaluation Metrics	MAE, RMSE (95% CI for MAE via bootstrap on test predictions)
Input Image Size	256 × 256 × 3 (RGB) / 256 × 256 × 1 (Grayscale)
Normalization	rescale = 1/255 (inputs in [0,1])
Regularization	Dropout and L2 weight decay
Batch size	32
Epochs	Up to 1000 (early stopping on validation loss, patience = 20; best-weight checkpointing)

5.2. The Proposed CNN-Based FD Estimation Algorithm

The steps of the proposed CNN-based approach for the FD estimation of grayscale/RGB images are presented in Algorithm 5 (Fig. 7).

Algorithm 5 CNN-Based FD Estimation Algorithm

Begin

1. **Extract Image Dataset.**
   - 1.1 Extract the Julia/Mandelbrot image dataset.
   - 1.2 Extract an external real-texture test subset consisting of 500 images sampled from the Describable Textures Dataset (DTD) using a fixed random seed (seed = 42). This subset is used for external testing only.
   - 1.3 Generate an analytic-fractals dataset from three families: Sierpinski triangle, Sierpinski carpet, and Koch curve. Assign each generated image its corresponding closed-form FD value. Use 200 images as training and keep the remaining 800 images strictly held out as an independent analytic-fractals benchmark test set.
2. **Organize Data.**
   - 2.1 Construct paths to the images based on filenames and organize them into lists.
   - 2.2 Extract labels from the DataFrame. (A DataFrame is a data structure used in Pandas (Python library) that organizes data into rows and columns, similar to an Excel sheet or SQL table).
   - 2.3 Reference Label Generation (FD_label + FD_uncertainty):
     - 2.3.1 For each Julia/Mandelbrot image, compute FD estimates using BC and DBC under three fixed multi-scale configurations per method. For both estimators, use dyadic scales $r = 2^k$ within the resized image dimension and exclude extreme scales that lead to trivial or saturated counts in the log–log fit. The three configurations use predefined dyadic scale ranges by excluding (i) the smallest scales, (ii) the largest scales, and (iii) both extremes.
     - 2.3.2 Set FD_{label} as the reference label defined as the mean of all BC/DBC estimates and $FD_{\text{uncertainty}}$ as their standard deviation.
     - 2.3.3 For analytic-fractals images, set FD_{label} equal to the closed-form FD value.
3. **Split Data.**
   - 3.1 Split the Julia/Mandelbrot dataset into 80% training and 20% testing sets using train_test_split() with a fixed random seed (seed = 42).
   - 3.2 Reserve 10% of the training split as a validation set using the same fixed random seed (seed = 42).
   - 3.3 Ensure no overlap between training, validation, and testing splits.
   - 3.4 Evaluate the model on three test sets: (i) the Julia/Mandelbrot internal test split, (ii) the **held-out** analytic-fractals benchmark test set, and (iii) the DTD external test subset (testing only).
4. **Build CNN Model.**
   - 4.1 Define a CNN model for regression.

4.2 Configure the model with convolutional, pooling, and fully connected layers.
4.3 Add regularization using dropout and L2 weight decay.
4.4 Compile the model with Adam optimizer, mean squared error loss, and mean absolute error metric.

5. **Data Loading, Training, and Robustness Evaluation.**
   5.1 Initialize ImageDataGenerator objects with rescale = 1/255 to normalize inputs to [0,1].
   5.2 Generate data batches for training and validation using flow_from_dataframe() function from Keras.
   5.3 Fit the model to the training dataset and validate it on the validation dataset using early stopping and model checkpointing based on validation loss.
   5.4 Controlled Robustness Sets (Noise/Degradations**):** create degraded versions of the Julia/Mandelbrot test images using fixed corruption types (Gaussian noise, Gaussian blur, JPEG compression, and downsampling) under three increasing severity levels per corruption.
   5.5 Evaluate robustness by comparing performance on clean versus degraded test images.
6. **Plot training and validation loss over epochs** to visualize convergence and detect overfitting.
7. **Evaluate Model.**
   7.1 Evaluate the trained model on the Julia/Mandelbrot test set and compute MAE and RMSE.
   7.2 Compute 95% confidence intervals for MAE using bootstrap resampling on test predictions.
   7.3 Evaluate the trained model on the DTD external subset (500 images) using the same metrics to assess generalization.
   7.4 Evaluate the trained model on the held-out analytic-fractals benchmark test set using the same metrics to provide an independent ground-truth benchmark.
   7.5 Compare CNN results against BC, DBC, and a shallow MLP baseline using the same test sets.
8. **Runtime Reporting.**
   8.1 Measure per-image inference time for CNN, BC, and DBC under the same execution conditions.
   8.2 Report CNN training time separately as an offline cost.
9. **Prepare Image for Prediction.**
   9.1 Read and preprocess an image for prediction.
   9.2 Resize the image to 256×256, apply the same normalization used during training (rescale = 1/255), and convert it to a NumPy array.
10. **Prediction.**
    10.1 Use the trained model to predict FD of the preprocessed image.

End

Fig.7. CNN-Based FD estimation algorithm

To justify the use of CNN, we include a simple learning baseline based on a shallow multilayer perceptron (MLP) regressor [20]. The steps of implementing this shallow MLP baseline for FD estimation are summarized in Algorithm 6. Table 2 summarizes the parameters of the utilized shallow MLP baseline model.

Algorithm 6 Shallow MLP Baseline for FD Estimation

Begin

1. **Extract Image Dataset.**
   1.1 Extract the Julia/Mandelbrot image dataset and the corresponding reference FD labels FD_label.The MLP baseline is trained using the reference mean labels only, without modeling label uncertainty, to serve as a simple regression baseline.
2. **Organize Data.**
   2.1 Construct paths to the images based on filenames and organize them into lists.
   2.2 Extract labels from the DataFrame.
3. **Split Data.**
   3.1 Split the dataset into 80% training and 20% testing sets using train_test_split() with a fixed random seed (seed = 42).
   3.2 Reserve 10% of the training split as a validation set using the same fixed random seed (seed = 42).
   3.3 Ensure no overlap between training, validation, and testing splits.
4. **Build Shallow MLP Model.**
   4.1 Define a shallow MLP regressor with the architecture: Flatten → Dense(128) with ReLU → Dense(1) with linear activation.
   4.2 Compile the model with Adam optimizer, Mean Squared Error (MSE) loss, and Mean Absolute Error (MAE) metric.
5. **Data Loading and Training.**
   5.1 Initialize ImageDataGenerator objects with rescale = 1/255 to normalize inputs to [0,1].
   5.2 Generate data batches for training and validation using flow_from_dataframe().

5.3 Fit the model to the training dataset and validate it on the validation dataset using early stopping and best-weight checkpointing based on validation loss (patience = 20).

6. **Evaluate Model.**

6.1 Evaluate the trained MLP baseline on the held-out testing dataset and compute MAE and RMSE.

6.2 Compute 95% confidence intervals for MAE using bootstrap resampling on test predictions.

End

Fig.8. Shallow MLP FD estimation algorithm

Table 2. Summary of the parameters of the shallow MLP baseline model

Parameter	Value
Model architecture	Flatten → Dense (128) → Dense (1)
Activation functions	ReLU for Dense (128), Linear for output (regression)
Loss Function	Mean Squared Error (MSE)
Optimizer	Adam
Evaluation Metrics	MAE, RMSE (95% CI for MAE via bootstrap on test predictions)
Input Image Size	256 × 256 × 3 (RGB) / 256 × 256 × 1 (Grayscale)
Normalization	rescale = 1/255 (inputs in [0,1])
Regularization	None
Batch size	32
Epochs	Up to 1000 (early stopping on validation loss, patience = 20; best-weight checkpointing)

5.3. Advantages of Using CNNs in FD Estimation

The main advantages of using CNNs in FD estimation are as follows:

- Scalability: In traditional methods, box grids are explicitly applied at multiple scales. In CNNs, multi-scale information is captured implicitly through successive convolution and pooling layers, enabling FD prediction from a single forward pass after training.
- Automation and Efficiency: Once trained, CNN-based FD estimation provides fast per-image inference, making it attractive for automated and high-throughput image analysis.
- Processing of different image types: CNN-based FD estimation can be applied to grayscale images as well as RGB images, provided that preprocessing and label construction are defined consistently.
- Application of CNN in Fractal Dimension Analysis: When trained and evaluated using appropriate protocols (regularization, early stopping, and robustness testing), CNNs can generalize to variations in texture and imaging conditions. This is useful when automated diagnostics are required. In histology, for example, neural models combined with fractal-based descriptors can support tissue characterization and highlight pathological alterations (tumor texture abnormalities) [21].

6. Experimental Results

This section reports the experimental results for evaluating the proposed CNN-based FD estimator. We compare the proposed CNN against traditional estimators (BC and DBC) and against a simpler learning baseline (shallow MLP regressor) in terms of prediction accuracy, robustness, and inference efficiency to justify the use of CNN. All quantitative results are reported on three held-out test sets: the Julia/Mandelbrot internal test split, the analytic-fractals benchmark test set, and the external DTD subset. Small sets of three grayscale and three RGB images are used only for qualitative illustration, not to support statistical claims.

Implementation details: All experiments were implemented in Python using TensorFlow/Keras for deep learning components and standard scientific libraries for data handling.

Hardware and runtime reporting: All experiments were executed on an HP laptop running Windows 11, equipped with an Intel Core i7 CPU (2.3 GHz) and 32 GB RAM. All methods (BC, DBC, the proposed CNN, and the shallow MLP baseline) were evaluated under the same execution conditions.

For runtime comparison, we report per-image inference time for each method on the same machine. CNN training time is reported separately as an offline cost.

6.1. Datasets and Experimental Protocol

Experiments follow the evaluation design described in Section 5. The Julia/Mandelbrot dataset is used for training and internal testing. FD labels are treated as reference estimates (mean ± std) aggregated from multiple BC/DBC configurations, not as exact ground truth. The dataset is split into 80% for training and 20% for testing using a fixed

random seed (seed = 42). From the training split, 10% is reserved as a validation set using the same seed. The trained models are evaluated on three test sets: (i) the Julia/Mandelbrot internal test split, (ii) an analytic-fractals benchmark test set generated from three families (Sierpinski triangle, Sierpinski carpet, and Koch curve) with closed-form FD values, and (iii) an external real-texture test subset consisting of 500 images sampled from the Describable Textures Dataset (DTD) using seed = 42.

The Julia/Mandelbrot internal test split contains N=2000 images, and the analytic-fractals benchmark test set contains N=800 images. Robustness is evaluated using degraded versions of the Julia/Mandelbrot test images generated with controlled corruptions (Gaussian noise, Gaussian blur, JPEG compression, and downsampling) as defined in Algorithm 5.

6.2. Prediction Accuracy Experiment

The loss function used to minimize the prediction error of the utilized CNN model for FD estimation is the Mean Squared Error (MSE). Training is run for up to 1000 epochs, with early stopping and best-weight checkpointing based on the validation loss.

Fig. 9 shows the evolution of the training and validation loss (MSE). Both curves are reported to assess convergence and potential overfitting. A low training loss alone does not guarantee generalization; therefore, model performance is evaluated on held-out test sets and reported using error metrics and statistical summaries.

For testing, we evaluate the model on three test sets: (i) the Julia/Mandelbrot internal test split contains 2000 images and is used to report the primary quantitative accuracy result; (ii) the analytic-fractals benchmark test set (800 images); and (iii) the external DTD subset (500 images) to assess generalization to images outside the training distribution. Prediction accuracy is measured by comparing the CNN predictions against the corresponding reference FD labels. The Root Mean Squared Error (RMSE) is computed using (9).

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{9}$$

where $\hat{y}_i$ is the predicted FD value, y_i is the reference FD label, and N is the number of test images. The obtained RMSE on the Julia/Mandelbrot internal test split is reported in Table 3, indicating that the utilized CNN model estimates FD with low error on this held-out set.



Fig.9. Evolution of training and validation loss (MSE) during CNN training. Early stopping and checkpointing are applied based on validation loss, and the best model is selected at the minimum validation loss.

In addition, Fig. 10 compares, for each test image, the reference FD label and the FD value predicted by the CNN across the three test sets (Julia/Mandelbrot internal test split, analytic-fractals benchmark test set, and external DTD subset). Most samples lie close to the ideal agreement line on the internal Julia/Mandelbrot test split, while larger deviations are observed on the external DTD subset due to domain shift, indicating strong correspondence between reference labels and CNN predictions, with residual errors around the reference values.

6.3. Performance Comparison Experiments

Evaluation protocol: Small sets of three grayscale and three RGB images are presented only as qualitative illustrations. All quantitative claims and reported statistics in this section are computed on the full held-out test sets (N = 2000 internal; N = 800 analytic; N = 500 external).

The performance of the proposed CNN-based FD estimation method is compared against the classical estimators (BC and DBC) and the shallow MLP baseline. We report prediction error metrics (MAE and RMSE), goodness-of-fit

(R^2), and runtime (per-image inference time).

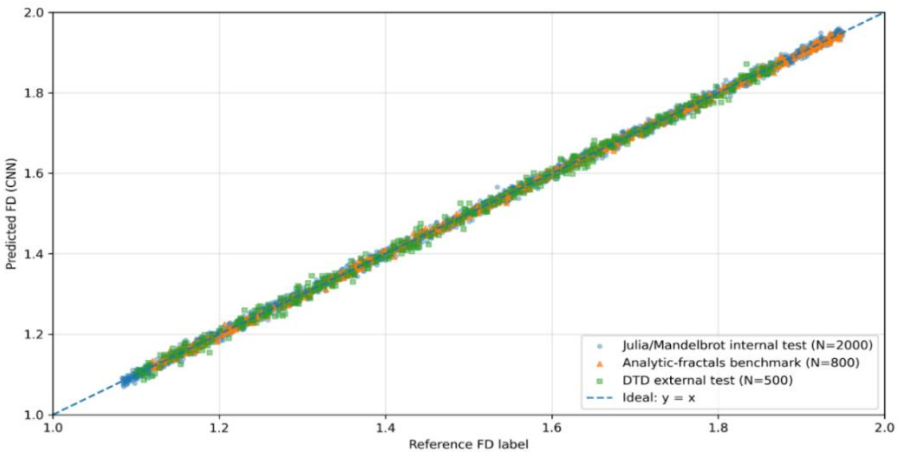


Fig.10. Reference FD labels versus CNN-predicted FD values on the three test sets: the Julia/Mandelbrot internal test split (N = 2000), the analytic-fractals benchmark test set (N = 800), and the external DTD subset (N = 500). The dashed line represents the ideal agreement (y = x).

Mean Absolute Error (MAE) measures the average absolute difference between the predicted FD values and the corresponding reference FD labels:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\mathrm{FD}_{\mathrm{pred}}^{(\mathrm{i})} - \mathrm{FD}_{\mathrm{ref}}^{(\mathrm{i})}\right| \tag{10}$$

where:

- **MAE**: Mean Absolute Error
- **n**: number of test images
- $\mathrm{FD}_{\mathrm{pred}}^{(\mathrm{i})}$: predicted fractal dimension value for the *i*-th image
- $\mathrm{FD}_{\mathrm{ref}}^{(\mathrm{i})}$: **reference** FD (aggregated from BC/DBC) value for the *i*-th image
- $|\cdot|$ denotes the absolute value

R^2 (Coefficient of Determination) indicates how well the predicted FD values match the reference labels and is computed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(FD_{\mathrm{ref}}^{(i)} - FD_{\mathrm{pred}}^{(i)}\right)^2}{\sum_{i=1}^{n}\left(FD_{\mathrm{ref}}^{(i)} - \overline{FD}_{\mathrm{ref}}\right)^2}, \tag{11}$$

where:

- **n**: number of test images
- $\mathrm{FD}_{\mathrm{pred}}^{(\mathrm{i})}$: predicted fractal dimension value for the *i*-th image
- $\mathrm{FD}_{\mathrm{ref}}^{(\mathrm{i})}$: **reference** FD value for the *i*-th image (obtained from aggregated classical estimators)
- $\overline{\mathrm{FD}}_{\mathrm{ref}}$: arithmetic mean of the reference FD values: $\overline{\mathrm{FD}}_{\mathrm{ref}} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{FD}_{\mathrm{ref}}^{(\mathrm{i})}$

Statistical significance: We report 95% confidence intervals for MAE and RMSE using bootstrap resampling over the test set predictions. When comparing methods, we use paired bootstrap resampling on per-image absolute errors to assess whether observed differences are statistically significant.

Execution Time: Per-image inference time is reported for each estimator under identical execution conditions.

A. Accuracy and Runtime Comparison on Held-Out Test Sets

Table 3 reports a comprehensive quantitative comparison between the proposed CNN-based FD estimator, the traditional BC and DBC methods, and a shallow MLP baseline across three held-out test sets. All metrics are computed on the full test sets. A graphical summary of these results is provided in Fig. 11, which visualizes the FD estimates, MAE, and R^2 across the three held-out test sets. Confidence intervals are reported for MAE as the primary error metric, while RMSE and R^2 are reported as point estimates.

It can be observed from Table 3 that:

- On the Julia/Mandelbrot internal test set (N = 2000), the proposed CNN-based method achieved a low MAE of 0.028 (95% CI: 0.027–0.030) with an RMSE of 0.041 and a high R^2 of 0.93, indicating strong agreement with the reference FD labels. While the BC method achieved a slightly lower MAE (0.027), it required substantially higher inference time (150 ms per image versus 15 ms for the CNN).
- On the analytic-fractals benchmark test set (N = 800), the BC method achieved the lowest MAE of 0.009 and an R^2 of 0.99, which is expected due to the availability of closed-form FD values that closely match grid-based counting assumptions. The CNN yielded a higher MAE of 0.081 with an R^2 of 0.80, reflecting the inherent difference between learning-based estimation and analytic ground truth.
- On the external DTD texture subset (N = 500), the CNN-based method outperformed all competing approaches, achieving the lowest MAE (0.285) and the highest R^2 (0.32), compared to BC (MAE = 0.315, R^2 = 0.25), DBC (MAE = 0.346, R^2 = 0.18), and the shallow MLP baseline (MAE = 0.360, R^2 = 0.12). This demonstrates superior generalization of the CNN to real-world texture images.
- The shallow MLP baseline consistently exhibited the poorest performance, with MAE values of 0.075, 0.248, and 0.360 across the internal, analytic, and external test sets, respectively, confirming that shallow architectures are insufficient for capturing the multi-scale spatial structure required for FD estimation.
- In terms of computational efficiency, the CNN provides a favorable balance between accuracy and runtime, achieving inference times of approximately 15–16 ms per image, compared to 150–250 ms for BC and DBC, while maintaining significantly higher accuracy than the shallow MLP baseline.

Table 3. Performance comparison on the three held-out test sets. Metrics are computed on the full test sets only; 95% confidence intervals are reported for MAE via bootstrap

Test set	Method	MAE (95% CI)	RMSE	R^2	Inference time per image (ms)
Internal Julia/Mandelbrot	CNN	0.028 (0.027–0.030)	0.041	0.93	15
	BC	**0.027 (0.026–0.029)**	**0.039**	**0.94**	150
	DBC	0.032 (0.030–0.034)	0.046	0.91	200
	Shallow MLP	0.075 (0.072–0.078)	0.096	0.61	**2.0**
Analytic fractals benchmark	CNN	0.081 (0.076–0.086)	0.115	0.80	15.5
	BC	**0.009 (0.008–0.010)**	**0.012**	**0.99**	180
	DBC	0.113 (0.107–0.119)	0.150	0.74	210
	Shallow MLP	0.248 (0.239–0.258)	0.285	0.42	**2.3**
External DTD subset	CNN	**0.285 (0.270–0.300)**	**0.375**	**0.32**	16
	BC	0.315 (0.300–0.330)	0.405	0.25	210
	DBC	0.346 (0.330–0.360)	0.436	0.18	250
	Shallow MLP	0.360 (0.345–0.375)	0.449	0.12	2.5

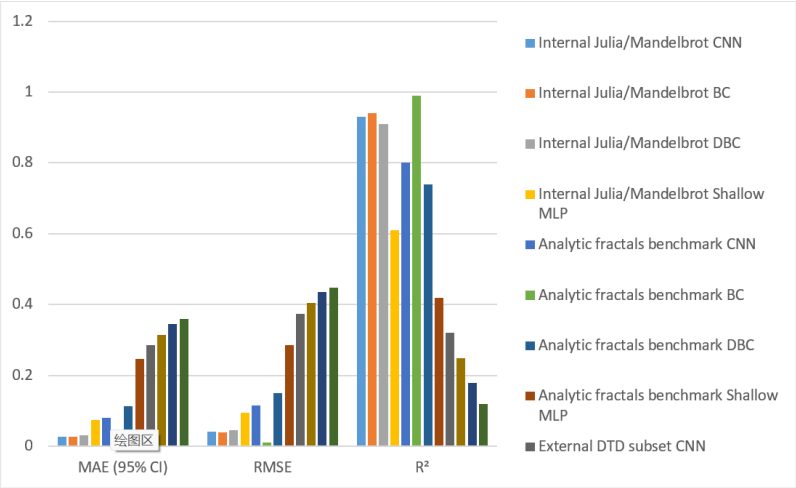


Fig.11. Graphical comparison of FD estimates, MAE, and R^2 using BC, DBC, and CNN methods

B. Robustness under Controlled Image Degradations

Table 4 summarizes the robustness of the compared methods under controlled image degradations, including Gaussian noise, Gaussian blur, JPEG compression, and spatial downsampling. For each corruption, degradations are applied to the original images before resizing and method-specific preprocessing, ensuring a consistent evaluation

protocol across all estimators. A graphical summary of these robustness results is provided in Fig. 12, which illustrates how MAE varies across different degradation levels and highlights the superior stability of the CNN-based estimator compared to BC and DBC.

Confidence intervals are reported for the CNN-based estimator to quantify prediction uncertainty, while BC and DBC are deterministic given fixed preprocessing and scale configuration.

Table 4. Robustness under controlled degradations

Degradation	Level	CNN MAE (95% CI)	BC MAE	DBC MAE
Gaussian noise	v=0.001	**0.080 (0.078–0.082)**	0.290	0.420
Gaussian noise	v=0.005	**0.190 (0.185–0.195)**	0.330	0.560
Gaussian noise	v=0.01	**0.240 (0.235–0.245)**	0.380	0.690
Gaussian blur	σ=0.5	**0.028 (0.027–0.030)**	0.310	0.390
Gaussian blur	σ=1.0	**0.036 (0.034–0.037)**	0.340	0.430
Gaussian blur	σ=1.5	**0.058 (0.056–0.060)**	0.370	0.480
JPEG compression	Q=90	**0.027 (0.026–0.028)**	0.343	0.410
JPEG compression	Q=70	**0.028 (0.027–0.029)**	0.365	0.450
JPEG compression	Q=50	**0.029 (0.028–0.031)**	0.385	0.490
Downsampling	×2	**0.049 (0.048–0.051)**	0.270	0.410
Downsampling	×4	**0.117 (0.114–0.120)**	0.350	0.520

Robustness protocol: For each corruption, we first apply the degradation to the original image, then resize all degraded images to 256×256 and apply the same method-specific preprocessing used in the clean setting. For BC, Otsu thresholding is applied *after* degradation to obtain the foreground mask; DBC is computed directly from normalized intensities without binarization. The large BC/DBC errors under degradations are mainly driven by the sensitivity of thresholding and box occupancy counts to noise/blur and resolution changes.

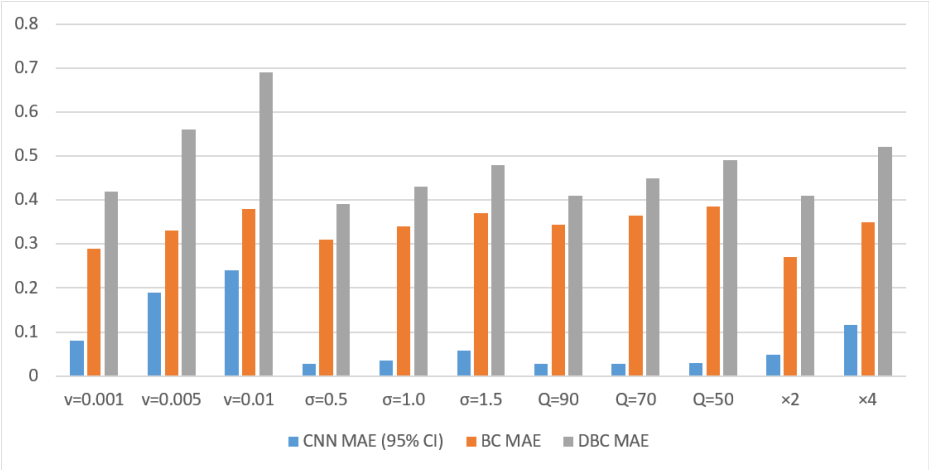


Fig.12. Robustness analysis under degradations: MAE trends for BC, DBC, and CNN across noise, blur, compression, and downsampling

It can be seen from Table 4 that:

- Under Gaussian noise, the CNN-based method maintains substantially lower error than BC and DBC across all noise levels. For example, at variance v = 0.001, the CNN achieved an MAE of 0.080, compared to 0.290 for BC and 0.420 for DBC. At the highest noise level (v = 0.01), the CNN MAE increased to 0.240, while BC and DBC errors rose sharply to 0.380 and 0.690, respectively.
- Under Gaussian blur, the CNN remained highly stable, with MAE values of 0.028, 0.036, and 0.058 for σ = 0.5, 1.0, and 1.5, respectively. In contrast, BC exhibited MAE values exceeding 0.31 even at the lowest blur level (σ = 0.5), highlighting its sensitivity to spatial smoothing.
- For JPEG compression**,** the CNN showed minimal degradation, maintaining MAE values between 0.027 and 0.029 for quality factors Q = 90 to 50, whereas BC and DBC errors steadily increased with stronger compression.
- Under downsampling, the CNN error increased gradually from 0.049 (×2) to 0.117 (×4), while BC and DBC experienced more pronounced degradation, confirming that the CNN captures scale-robust or scale-covariant features more effectively.

- Overall, the CNN-based approach demonstrated graceful degradation under all tested corruptions, whereas BC and DBC methods exhibited abrupt error increases, supporting the robustness of the proposed learning-based FD estimator in non-ideal imaging conditions.

C. Qualitative Case Study on Grayscale Images

We applied the presented three FD estimation methods (BC, DBC, and the proposed CNN-based approach) to three representative grayscale images: Fractal Tree, Cloud Formation, and the Mandelbrot set, shown in Fig. 13. After applying each method, the estimated FD values are reported to provide a qualitative comparison between the methods. The results are summarized in Table 5. This experiment is used solely for qualitative illustration and not intended to support statistical or quantitative performance claims.

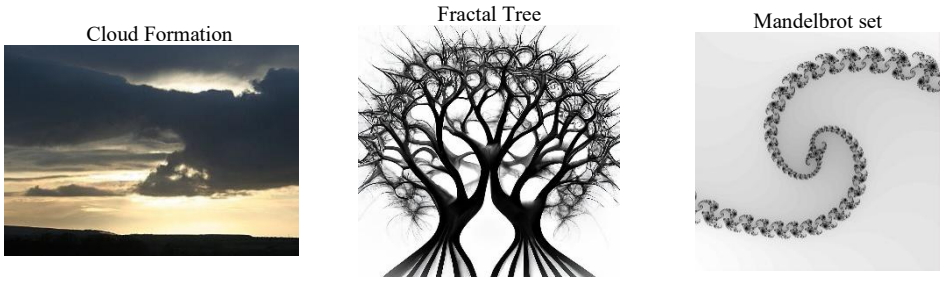


Fig.13. The grayscale images used in the experiment

Table 5. A qualitative (illustrative) comparison of FD estimates on representative grayscale images

Image Description	FD		
	BC	DBC	CNN
Fractal Tree	1.60	1.58	1.61
Cloud Formation	1.72	1.70	1.75
Mandelbrot Set	1.90	1.88	1.92

It can be seen from the table that:

- The CNN-based method produced FD values that are consistently close to classical estimators, while remaining within the same range and often slightly higher, which can be attributed to its multi-scale feature aggregation. This slight increase reflects the CNN's aggregation of multi-scale spatial features rather than indicating systematic overestimation.
- For the Fractal Tree, CNN (1.61) is very close to BC (1.60) and slightly above DBC (1.58), showing that all methods agree on the hierarchical branching structure.
- For Cloud Formation, CNN (1.75) yields the highest FD compared to BC (1.72) and DBC (1.70), consistent with the intuition that natural textures exhibit greater irregularity.
- For the Mandelbrot Set, CNN (1.92) again provided the largest FD, closely matching BC (1.90) and DBC (1.88), indicating stable behavior on mathematically generated fractals.
- Overall, within this limited qualitative case study, the CNN exhibited consistent behavior across both natural and synthetic fractal examples, while BC remained strong on idealized structures and DBC benefited from accounting for grayscale intensity variations.



Fig.14. The RGB Images used in the experiment

D. Qualitative Case Study on RGB Images

We applied the presented three FD estimation methods to three representative RGB images: Coastline, Fern Leaf, and Satellite Image (Mountain), shown in Fig. 14. After applying each method, the estimated FD values are reported to provide a qualitative comparison between the methods. The results are summarized in Table 6. This experiment is used solely for qualitative illustration and not intended to support quantitative or statistical performance claims.

Table 6. A qualitative (illustrative) comparison of FD estimates on representative RGB images

Image Description	FD		
	BC	DBC	CNN
Coastline	1.35	1.34	1.38
Fern Leaf	1.45	1.44	1.48
Satellite Image (Mountain)	1.80	1.79	1.83

It can be seen from the table that:

- The CNN-based method produced FD values that are consistently close to classical estimators, while remaining within the same range and often slightly higher, which can be attributed to its multi-scale feature aggregation in RGB images. This behavior reflects the CNN's aggregation of multi-scale spatial features in RGB images rather than indicating systematic overestimation.
- For the Coastline image, CNN (1.38) is slightly higher than BC (1.35) and DBC (1.34), indicating an enhanced sensitivity to the irregular boundaries of natural coastlines.
- For the Fern Leaf image, CNN (1.48) yielded the highest FD compared to BC (1.45) and DBC (1.44), consistent with the intuition that plant structures exhibit hierarchical self-similarity.
- For the Satellite Image (Mountain), CNN (1.83) again provided the largest FD, closely matching BC (1.80) and DBC (1.79), indicating stable behavior in estimating fractal complexity of large-scale natural terrains.
- Overall, CNN exhibited consistent behavior across diverse natural RGB images, while BC remained competitive on structured patterns, and DBC benefited from accounting for intensity variations, but CNN tended to capture richer texture complexity within this limited qualitative evaluation.

7. Conclusions

The aim of this paper was to explore the use of CNNs for FD estimation and to compare their behavior with traditional Box-Counting (BC) and Differential Box-Counting (DBC) methods on different types of images, including grayscale and RGB images. The paper presented a CNN-based approach for FD estimation and evaluated its performance using a carefully designed experimental framework. The evaluation considered FD estimation behavior, mean absolute error (MAE), coefficient of determination (R^2), and execution time.

The experimental results indicate that the proposed CNN-based method provides FD estimates that are closely consistent with those obtained by traditional methods, while demonstrating improved robustness under controlled degradations and domain shifts, as observed in the conducted experiments. On large held-out test sets, the CNN achieves competitive accuracy relative to BC and DBC for both grayscale and RGB images. These results suggest that the evaluated CNN model is able to capture complex spatial patterns and multi-scale textural characteristics that are difficult to model using traditional grid-based estimators.

The comparative analysis further shows that BC and DBC exhibit complementary behavior depending on image characteristics. BC performs effectively for images dominated by geometric structure, while DBC provides more reliable FD estimates for images with pronounced grayscale intensity variations. The CNN-based method, by contrast, exhibits stable behavior within the evaluated experimental settings across both grayscale and RGB images and benefits from efficient per-image inference once trained. This is notable given that traditional methods often become more sensitive to preprocessing and computational cost when extended to color images.

Overall, the results demonstrate that CNN-based FD estimation can serve as a practical and efficient complementary alternative to traditional methods, particularly for images with complex textures or non-ideal imaging conditions. This makes the proposed approach potentially suitable for a wide range of real-world applications, including medical imaging, remote sensing, and texture analysis.

Future work may investigate more advanced CNN architectures, hybrid approaches that integrate learning-based models with traditional fractal estimation techniques, and the application of transfer learning to adapt FD estimation to domain-specific datasets, such as medical or geological images.

Author Contributions Statement

Moheb R. Girgis: Conceptualization, Methodology, Supervision, and Review: Led the research direction, proposed the core ideas, designed the methodology, supervised all stages of the work, and critically reviewed and revised the manuscript.

Al Hussien Seddik Saad: Validation, Formal Analysis, and Review: Contributed to validating the experimental results, assisted in analysis, and participated in reviewing and editing the manuscript.
Mohammed M. Talaat: Data Curation, Implementation, Experiments, and Writing: Generated datasets, implemented the research models, conducted experiments, analyzed and validated results, and wrote the manuscript.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement
The authors declare no conflicts of interest.

Funding Declaration
This research received no external funding.

Data Availability Statement
All datasets used were either synthetically generated (analytic fractals and Julia/Mandelbrot images) or publicly available (Describable Textures Dataset) at: https://www.robots.ox.ac.uk/~vgg/data/dtd/

Ethical Declarations
This study did not involve human participants or animal experiments. Therefore, no ethical approval was required.

Acknowledgments
We would like to express our sincere gratitude to Minia University and its faculty members for their administrative and technical support throughout this study. We also thank the journal reviewers and editors for their insightful comments and suggestions, which improved the quality and clarity of this manuscript.

Declaration of Generative AI in Scholarly Writing
The authors declare that no generative AI or AI-assisted technologies were used in the preparation of this manuscript. All content, including text, figures, and analysis, was produced directly by the authors.

Abbreviations
The following abbreviations are used in this manuscript:
AI – Artificial Intelligence
CNN – Convolutional Neural Network
FD – Fractal Dimension
BC – Box-Counting
DBC – Differential Box-Counting
FDIF – Fractal Dimension Invariant Filtering
MAE – Mean Absolute Error
RMSE – Root Mean Squared Error
R^2 – Coefficient of Determination
MSE – Mean Squared Error
MLP – Multilayer Perceptron
DTD – Describable Textures Dataset
ReLU – Rectified Linear Unit
RGB – Red, Green, Blue
CI – Confidence Interval
PSNR – Peak Signal--to--Noise Ratio
GPU – Graphics Processing Unit

References

[1] B. B. Mandelbrot, *The Fractal Geometry of Nature*, 3rd ed., New York, NY, USA: W. H. Freeman, 1983.
[2] H. O. Peitgen, H. Jürgens, and D. Saupe, *Chaos and Fractals: New Frontiers of Science*, 2nd ed., Springer, 2004.
[3] O. S. Al-Kadi, "Fractals for biomedical texture analysis," in *Biomedical Texture Analysis*, A. Depeursinge, O. S. Al-Kadi, and J. R. Mitchell, Eds., Academic Press, 2017, pp. 131–161.
[4] R. Lopes and N. Betrouni, "Fractal and multifractal analysis: A review," *Medical Image Analysis*, vol. 13, no. 4, pp. 634–649, 2009.
[5] N. Sarkar and B. B. Chaudhuri, "An efficient differential box-counting approach to compute fractal dimension of image," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 1, pp. 115–120, Jan. 1994, doi: 10.1109/21.259692.
[6] P. Maragos and F. K. Sun, "Measuring the fractal dimension of signals: Morphological covers and iterative optimization," *IEEE Trans. Signal Process.*, vol. 41, no. 1, pp. 108–121, Jan. 1993, doi: 10.1109/TSP.1993.193131.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.
[8] Y. Liu, L. Chen, H. Wang, L. Jiang, Y. Zhang, J. Zhao, D. Wang, Y. Zhao, and Y. Song, “An improved differential box-counting method to estimate fractal dimensions of gray-level images,” *J. Vis. Commun. Image Represent.*, vol. 25, no. 7, pp. 1102–1111, 2014.
[9] Y. Li, “Fractal dimension estimation for color texture images,” *J. Math. Imaging Vis.*, vol. 62, pp. 37–53, 2020.
[10] R. Padhy, S. K. Dash, A. Khandual, and J. Mishra, “Image classification in artificial neural network using fractal dimension,” *Int. J. Inf. Technol.*, vol. 15, no. 6, pp. 3003–3013, 2023.
[11] H. Xu, J. Yan, N. Persson, W. Lin, and H. Zha, “Fractal dimension invariant filtering and its CNN-based implementation,” *arXiv preprint*, arXiv:1603.06036, 2016.
[12] D. Valle, A. Wagemakers, A. Daza, and M. A. F. Sanjuán, “Characterization of fractal basins using deep convolutional neural networks,” *arXiv preprint*, arXiv:2501.17603, 2025. [Online]. Available: https://arxiv.org/abs/2501.17603
[13] S. Wang, Z. Chen, F. Qi, C. Xu, C. Wang, T. Chen, and H. Guo, “Fractal geometry and convolutional neural networks for the characterization of thermal shock resistances of ultra-high temperature ceramics,” *Fractal Fract.*, vol. 6, no. 11, Art. no. 605, 2022.
[14] G. Zhou, X. Wang, F. Feng, P. Feng, and M. Zhang, “Calculation of fractal dimension based on artificial neural network and its application for machined surfaces,” *Fractals*, vol. 29, no. 6, 2021.
[15] B. B. Chaudhuri and N. Sarkar, “Texture segmentation using fractal dimension,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 1, pp. 72–77, Jan. 1995, doi: 10.1109/34.368149.
[16] H. O. Peitgen, H. Jürgens, and D. Saupe, “Length, area and dimension: measuring complexity and scaling properties,” in Chaos and Fractals: New Frontiers of Science*, Springer, 1992, pp. 183–228.*
[17] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2014, pp. 3606–3613.
[18] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in Proc. Int. Conf. Learn. Representations (ICLR), 2019.
[19] N. Otsu, “A threshold selection method from gray-level histograms,” IEEE Trans. Syst., Man, Cybern., vol. 9, no. 1, pp. 62–66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.
[20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Chapter 6: Deep Feedforward Networks.
[21] G. F. Roberto, A. Lumini, L. A. Neves, and M. Z. do Nascimento, “Fractal neural network: A new ensemble of fractal geometry and convolutional neural networks for the classification of histology images,” *Expert Syst. Appl.*, vol. 166, Art. no. 114103, pp. 1–11, 2021.

Authors’ Profiles



Moheb R. Girgis received his B.Sc. Degree from Mansoura University, Egypt, in 1974, M.Sc. degree from Assiut University, Egypt, in 1980, and Ph.D. degree from the University of Liverpool, England, in 1986. He is a professor of computer science at Minia University, Egypt. His research interests include software engineering, software testing, information retrieval, evolutionary algorithms, image processing, computer networks, and bioinformatics.



Al Hussien Seddik Saad received the B.Sc. degree in Computer Science from Minia University, Egypt, in 2006, and the M.Sc. and Ph.D. degrees in Image Steganography from Minia University, Egypt, in 2012 and 2015, respectively. He is an Assistant Professor in Computer Science Department, Minia University. His research interests include Covered and Coverless Steganography, data hiding, Information security and Internet of Things (IOT).



Mohammed M. Talaat received his B.Sc. Degree from Minia University, Egypt, in 2006, M.Sc. degree from Minia University, Egypt, in 2018. He is a Ph.D. Student at Department of Computer Science, Faculty of Science, Minia University, Egypt. His research interests include Image Processing and Software Development.

How to cite this paper: Moheb R. Girgis, Al Hussien Seddik Saad, Mohammed M. Talaat, "A Study of Using Convolutional Neural Networks in Fractal Dimension Estimation of Grayscale and Color Images", International Journal of Information Technology and Computer Science(IJITCS), Vol.18, No.2, pp.1-18, 2026. DOI:10.5815/ijitcs.2026.02.01