Article

Fuzzy Edge-Detection as a Preprocessing Layer in Deep Neural Networks for Guitar Classification

Cesar Torres, Claudia I. Gonzalez * and Gabriela E. Martinez

Tijuana Institute of Technology/TECNM, Tijuana 22414, Mexico
* Correspondence: cgonzalez@tectijuana.mx

Abstract: Deep neural networks have demonstrated the capability of solving classification problems using hierarchical models, and fuzzy image preprocessing has proven to be efficient in handling uncertainty found in images. This paper presents the combination of fuzzy image edge-detection and the usage of a convolutional neural network for a computer vision system to classify guitar types according to their body model. The focus of this investigation is to compare the effects of performing image-preprocessing techniques on raw data (non-normalized images) with different fuzzy edge-detection methods, specifically fuzzy Sobel, fuzzy Prewitt, and fuzzy morphological gradient, before feeding the images into a convolutional neural network to perform a classification task. We propose and compare two convolutional neural network architectures to solve the task. Fuzzy edge-detection techniques are compared against their classical counterparts (Sobel, Prewitt, and morphological gradient edge-detection) and with grayscale and color images in the RGB color space. The fuzzy preprocessing methodologies highlight the most essential features of each image, achieving favorable results when compared to the classical preprocessing methodologies and against a pre-trained model with both proposed models, as well as achieving a reduction in training times of more than 20% compared to RGB images.

Keywords: convolutional neural networks; guitar recognition; fuzzy edge-detection

1. Introduction

Artificial intelligence (AI) allows us to automate multiple tasks in different areas, such as artificial vision, which makes it possible to perform several jobs that were previously believed only humans would be capable of doing. The tasks are those such as object identification or object location in digital images; processes such as these require previous knowledge of the objects you are looking for, as well as the context in which they are found, requirements that are completely related to the vision and cognitive abilities of humans.

In recent years, deep neural networks (DNN) have made a huge advances in recognition tasks in multiple areas due to the capabilities of feature-extraction layers that DNNs have embedded in their design, which make them very attractive in multiple disciplines. Zhang et al. [1] created a fruit classification system using a deep neural network to replace handcrafted features, beating state-of-the-art approaches. Horng et al. [2] used a DNN, feeding it with aerial images to classify tree areas and help in the understanding of land use. Sebti et al. [3] provided a solution for forecasting and diagnosing of diabetic retinopathy by training a convolutional neural network with retina images, achieving over 96% accuracy.

Previous authors tackled the guitar/instrument classification problem, for example, Rabelo et al. [4] used a support vector machine to classify the sounds of guitars (in a binary problem) from two brands, obtaining 95% accuracy with a particular approach. Yet, this was done by using the sound (music notes) generated by the instrument, instead of images, to perform the classification tasks. Banerjee et al. [5] used musical notes and support vector machines to classify string instruments (cello, double bass, guitar, and violin), obtaining up to 100% recognition with random forest. This paper proposes an
artificial vision system that performs the classification by body types of some of the most popular guitar styles found nowadays. The system operates by exploiting the capabilities of the feature-extraction layers of DNN, feeding them with pictures of the guitars that need to be classified by the user. Given the lack of previous attempts to do so, we created a dataset by scraping images from popular search engines to generate a competent database that contained sufficient images of some of the most popular guitar models. The application of image-based instrument-classification systems could have great relevance in retail stores and warehouses of music instrument franchises, where there is a constant flow of inventory involved, especially of used products. Intelligent systems could be applied for inventory keeping or organization of unlabeled items. Another form of implementation viable for the music industry is a complementary system for off-site quoting services for instrument purchases or store credit requests, making it more compelling for customers to sell used instruments to the store for resale.

The process of detecting edges is an essential part of pattern-recognition systems; it simplifies the analysis by reducing the image to its minimal expression, and by doing so, it reduces the amount of processing needed [6]. This could be considered a difficult task, especially when the images contain noise or include irrelevant information. To solve this challenge, some fuzzy edge-detectors have been proposed. In Tao et al. [7], Hu et al. [8], and Ontiveros-Robles et al. [9], the authors presented some edge-detection methodologies with their foundation in type-1 fuzzy systems, T1FS. In Mendoza et al. [10,11], the authors propose fuzzy edge-detection based on the Sobel operator and interval type-2 fuzzy systems IT2FS. In Biswas et al. [12] and Melin et al. [13], the Canny method and the morphological gradient approach were improved using IT2FS, respectively. In the area of pattern-recognition, fuzzy edge-detection methods play an important role in improving the recognition rate when comparing the results with images without processing or when traditional filters are applied. We can mention other research works, where some edge-detection methodologies were based on T1FS, IT2FS, and general type-2 fuzzy sets (GT2FS) used in the preprocessing pipeline for face-recognition based on a monolithic neural network [14,15]. Mendoza et al. [16] applied edge-detectors to two face databases; this detection system was based on T1 and IT2FS. The edges found were used as inputs for a face-recognition system; the authors concluded that the recognition achieved by the system was improved when fuzzy edge-detection methods were applied. Martinez et al. [17] presented a face-recognition method with its foundation in modular neural networks, with favorable results when the fuzzy Sobel edge-detection was performed.

The success of convolutional neural networks (CNNs) in classification is due to their ability to perform both feature-extraction and classification, and most models perform very well without preprocessing steps. However, sometimes, the dataset images are unbalanced, have lower resolution, poor quality, or acquired some noise or uncertainty during the capture process. Due to these facts, some approaches use additional preprocessing methods, including image resizing, data augmentation, cropping, converting to grayscale to reduce the preprocessing time, and adding filters and equalization to improve the image quality or resolution. In the literature, we can find some works that have shown that the use of preprocessing methods before CNN-based models improves the results. Cordero-Martinez et al. [18,19] presented a comparison of four image preprocessing methods to classify diabetic retinopathy using CNN, and the authors evidenced and concluded that the preprocessing steps are important to increase the accuracy of the results. Kato et al. [20] proposed a preprocessing approach applied before a CNN-based model, and the results were 34.8% higher than the conventional CNN model. In Musallam et al. [21], three preprocessing steps were proposed to enhance the quality of MRI (magnetic resonance imaging) before these were introduced to a deep convolutional neural network (DCNN), which involved removing confusing objects, using a non-local mean algorithm (NLM), and applying histogram equalization. These were applied for automatic detection of brain tumors in MRI images and experimental results proved the robustness of the proposed architecture, which increased the detection accuracy of a variety of brain
diseases in a short time compared to other well-known deep convolutional neural network (DCNN) models such as VGG16 [22], VGG19 [22], and hybrid CNN-SVM [23]. Finally, Lăzărescu et al. [24] presented an algorithm for fingerprint classification using a combination of edge features and CNN. In this work, some preprocessing steps were applied, including edge-enhancement operations, data resizing, data augmentation, and the images were enhanced using Prewitt and Laplacian of Gaussian. The proposed algorithm achieved a very good performance compared to the traditional hand-crafted features.

The motivation to implement fuzzy edge-detection in this paper as a preprocessing phase was that fuzzy logic is a good technique to model the uncertainty or noise encountered in images where, with an appropriate filtering operator, this will be suppressed. The fuzzy methodologies (Prewitt, Sobel, and morphological gradient fuzzy edge-detectors) presented in this paper consider implementing only T1 fuzzy sets and are referenced from the previous state-of-the-art research [25,26]. The implementation and combination of fuzzy preprocessing techniques and convolutional neural networks produce powerful artificial intelligence tools for pattern recognition.

2. Materials and Methods

2.1. Theoretical Concepts

2.1.1. Convolutional Neural Network

The convolutional neural network, also known as CNN, is one of the most popular deep-learning techniques. where a computer system imitates the visual cortex and brain to identify patterns from visual elements [27]. CNNs have been used in multiple fields, such as medical [28–31], autonomous vehicles [32–34], and agricultural [35,36], just to name a few. A CNN differs from a conventional neural network (NN) because it contains at least a convolutional layer. The basic architecture of a CNN consists of the following layers: input, convolutional, activation fn., pooling, dense, and output layer.

1. **Input layer**: The input layer constitutes the input of the data that are going to be fed to the network, with a fixed size;
2. **Convolutional layer**: This layer applies a scalar product with a small matrix (kernel) to extract features or patterns from the input. generating a new matrix with the extracted contents;
3. **Activation function**: CNNs use activation functions to trigger the neurons when the individual values reach a threshold; these functions allow us to normalize the input data and perform non-linearization of the data. One of the most used activations is a rectified linear unit (ReLu); the output of the activation of a ReLu is the maximum between the values of the input;
4. **Pooling layer**: This layer allows us to create a smaller representation of the input data, and by doing so, we can reduce the number of features held by the input. The maximum pooling layer is one of the most used layers; this layer uses a filter with size $n \times n$, which does a pass all over the input, as the convolution layer. In this case, the maximum value of this filter is stored as the output in the new representation, reducing the dimensions of the input proportionally to the dimensions of the filter and its stride;
5. **Dense layer (fully connected)**: Consists of the weights and biases (like a traditional feed-forward neural network). The input of these layers is the output of a convolution or pooling layer;
6. **The output layer**: This layer contains the neurons that will provide the output of the input that was fed through the model.

2.1.2. Fuzzy Logic Systems

Fuzzy logic aims to perform representations in nonlinear ways as human logic does. Often, linguistic terms are used that differ from conventional logic where it is usually binary, and fuzzy logic allows gradual representations in a continuous space, thereby allowing levels of uncertainty. Zadeh introduced the fuzzy sets given the discontent of classic sets
(crisp sets) [37]. Fuzzy sets allow the use of membership, meaning their elements are capable of being part of one or more classes at the same time. The range of these sets is defined as human logic, where they depend on the concept or user applying them.

The T1FS set $A$ is from Universe $X$ that goes from $[0, 1]$, belonging to a continuous function, i.e., $\mu_A : X \rightarrow [0, 1]$. The T1 membership function $A$ is denoted as $\mu_A(x)$; this function is defined in Equation (1).

$$A = \{(x, \mu_A(x)) | x \in X\}$$  

The different membership functions (MFs) most used to represent a fuzzy set are the following: the triangular MF, the trapezoidal MF, and the Gaussian MF, which is used in the fuzzy edge-detection approach presented in this paper. This last consists of two parameters $\{c, \sigma\}$ and is expressed in Equation (2); variable $c$ represents the mean of the MF and $\sigma$ the amplitude.

$$\text{gaussian}(x; c, \sigma) = \exp\left(-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2\right)$$  

A fuzzy inference system (FIS) is based on if-then rules with fuzzy reasoning [38] applied to fuzzy sets. The basic structure of a FIS constitutes a database, along with a reasoning mechanism, which infers a reasonable conclusion from the inputs, outputs, and the knowledge provided by the database. Popular fuzzy inference systems are: Mamdani [39], Tsukamoto, and Takagi-Sugeno-Kang [40].

2.1.3. Edge-Detection Methods

Image-processing techniques consist of the manipulation of digital images to hide or highlight details, specific patterns, improve illumination, or eliminate noise caused by external agents such as artifacts that can be caused by the camera sensor or movement when taking the picture. This processing consists of applying an operation in a pixel window (kernel), which goes through the images, changing their content uniformly to create a new image [41]. Equation (3) describes the process of the application of a kernel to an image, where the $k$ represents the kernel, $r, c$ represents the coordinates of the value of the kernel, where $r$ represents the row and $c$ the column, and the variable $f$ represents the input image.

$$g(x, y) = \sum_{r=-a}^{a} \sum_{c=-b}^{b} k(r, c)f(x + r, y + c)$$  

Operations such as edge-detection are performed to reduce the amount of information that an image contains. Edge-detection techniques identify discontinuities in the brightness levels of the image to identify borders.

Among the most popular edge-detection techniques are the Roberts, Sobel, and Prewitt operators [42–44], which are focused on calculating the gradient with the first derivate of an image by applying a convolution operation that approximates the gradient and returns the first derivate for the horizontal and vertical directions.

Prewitt and Sobel Edge-Detection Methods

The traditional Prewitt and Sobel operators work in almost the same way; they both consist of a $3 \times 3$ neighborhood gradient operator, but they differ in the mask used in the convolutional process. The masks of the convoluted Prewitt operator on a grayscale image are defined by Equations (4) and (5), which correspond to Prewittx and Prewitty, respectively. In contrast, the masks used in the Sobel operator are expressed in Equation (6) (Sobelx) and Equation (7) (Sobely).

$$\text{Prewittx} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$  

Prewitt\(_x\) = \[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1
\end{bmatrix}
\] (5)

Sobel\(_x\) = \[
\begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\] (6)

Sobel\(_y\) = \[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\] (7)

The filter applies two different kernels to an image to generate the gradients \(g_x\) in Equation (8) for a horizontal orientation and \(g_y\) in Equation (9) for a vertical orientation. The coordinates of the input \(f\) are demonstrated in Figure 1, where the axis is represented by \(x\) for the horizontal axis and \(y\) for the vertical axis, and \(f\) for the image (input) source [26]. In Equations (7) and (8), kernel\(_x\) represents the mask Prewitt\(_x\) or Sobel\(_x\), and kernel\(_y\) is the mask for Prewitt\(_y\) or Sobel\(_y\), depending on the filter to be calculated.

\[
g_x = \sum_{i=1}^{3} \sum_{j=1}^{3} \text{kernel}_{i,j} \times f_{x+i-2,y+j-2} \tag{8}
\]

\[
g_y = \sum_{i=1}^{3} \sum_{j=1}^{3} \text{kernel}_{i,j} \times f_{x+i-2,y+j-2} \tag{9}
\]

| \(x-1,y-1\) | \(x-1,y\) | \(x-1,y+1\) |
|-------------|-------------|-------------|
| \(x,y-1\)   | \(x,y\)     | \(x,y+1\)  |
| \(x+1,y-1\) | \(x+1,y\)   | \(x+1,y+1\) |

**Figure 1.** Coordinates of the input image \(f\).

The magnitude of the gradients is obtained with Equation (10), which takes into consideration the results of the calculations of \(g_x\) and \(g_y\) from the image \(f\) using Equations (8) and (9); these convolutions are applied using the kernels defined in Equations (4)–(7).

\[
G[f(x,y)] = \sqrt{g_x^2 + g_y^2} \tag{10}
\]

**Morphological Gradient (MG)**

MG is an edge-detection technique that calculates the first derivative of an image in the four orientations of the image, i.e., vertical, horizontal, and its diagonals (0\(^\circ\), 45\(^\circ\), 90\(^\circ\), and 135\(^\circ\)). This is demonstrated in Figure 2a, where the gradients are indicated by the following variables \(G_1\), \(G_2\), \(G_3\), and \(G_4\). The calculation of the gradients is made in the following way.

\[G_l = 1 \ldots 4\text{ represents the direction of the edge (gradient). It is calculated with Equation (11) by using a } 3 \times 3 \text{ kernel. In Equation (11), } z_l \text{ represent the coefficient of each of the matrix positions shown in Figure 2b, where the coefficients are calculated with Equation (12); } f \text{ is the representation of the image, i.e., the x-axis for the columns, and y-axis for the rows. The edge value is denoted with the variable “Edges”, which is integrated using Equation (13) [13,25,45].} \]
The purpose of this investigation was to compare the effect of performing diverse image preprocessing techniques before feeding the images into a CNN to perform a classification test. Among the techniques performed were fuzzy edge-detection filters. Fuzzy filters (in this case edge-detection filters) allow us to determine the membership of a pixel to a boundary or a uniform region of the image, taking in consideration uncertainty. The general methodology of the proposal is illustrated in Figure 3 and explained as follows.

\[ G_1 = \sqrt{(z_5 - z_2)^2 + (z_5 - z_8)^2}, \quad G_2 = \sqrt{(z_5 - z_4)^2 + (z_5 - z_6)^2}, \quad G_3 = \sqrt{(z_5 - z_1)^2 + (z_5 - z_9)^2}, \quad G_4 = \sqrt{(z_5 - z_3)^2 + (z_5 - z_7)^2} \]
\[ z_1 = f(x - 1, y - 1), \quad z_2 = f(x, y - 1), \quad z_3 = f(x + 1, y - 1), \quad z_4 = f(x - 1, y), \quad z_5 = f(x, y), \quad z_6 = f(x + 1, y), \quad z_7 = f(x - 1, y + 1), \quad z_8 = f(x, y + 1), \quad z_9 = f(x + 1, y + 1) \]
\[ \text{Edges} = G_1 + G_2 + G_3 + G_4 \]

2.2. Methodology

The purpose of this investigation was to compare the effect of performing diverse image preprocessing techniques before feeding the images into a CNN to perform a classification test. Among the techniques performed were fuzzy edge-detection filters. Fuzzy filters (in this case edge-detection filters) allow us to determine the membership of a pixel to a boundary or a uniform region of the image, taking in consideration uncertainty. The general methodology of the proposal is illustrated in Figure 3 and explained as follows.

2.2.1. Input Database

In this study case, we used a self-made dataset that contained a balanced distribution of images of six different guitar styles (acoustic guitar, double cut, Les Paul, telecaster, stratocaster, and ukulele). Each class had a high degree of similarity with another, as denoted by the following pairings: acoustic with ukulele, double cut with Les Paul, and telecaster with stratocaster. These similarities only applied to our object of interest, while the images contained high levels of variation between each other. Since the images were scraped from the web by using popular search engines, the results included stock images for product advertisements from popular brands [46–48], or pictures posted in forums by owners of the instruments. The variations included a difference in position, orientation, and rotation of the object, illumination, other objects present in the frame (hands, cases,
stands, straps, or multiple guitars in the background), and resolution. The guitar database created consisted of 5400 images in color format (see Table 1), with the classes balanced (900 images per class), and with normalized dimensions of 150 × 150 pixels. The process of creating the dataset was as follows:

1. Create a list of keywords to perform a search in the search engines (Google Images and Bing Images); some of the keywords used are the follows:
   - Example of keywords used for the Les Paul class: “Gibson Les Paul, Les Paul style guitar, LP Junior, Les Paul Studio, Epiphone Les Paul, Les Paul, Les Paul copy, Ibanez Les Paul body guitar, ESP Les Paul, Burny Les Paul”;
   - Example of keywords used for Telecaster class: “Fender telecaster, Squier Tele, telecaster, telecaster type guitar, telecaster style guitar, tele guitar, telecaster guitar, Fender telecaster USA, telecaster custom shop”;
2. Download the top 100 hits from each keyword and save the images with a fixed size (150 × 150 px);
3. Manually prune the dataset to remove duplicates or outliers.

Table 1. Guitar database description.

| Name             | Description |
|------------------|-------------|
| Total images     | 5400        |
| Training images  | 4320        |
| Test images      | 1080        |
| Image size       | 150 × 150   |
| Data base format | JPG         |

2.2.2. Fuzzy Preprocessing Phase

This section contains an explanation of the membership function used, as well as the fuzzy rules and the methodology for the implementation of the fuzzy edge-detection approaches.

Fuzzy Sobel and Fuzzy Sobel Edge-Detection

The process to generate the fuzzy Sobel and fuzzy Prewitt edge-detection methods is similar, depending on the operator to be applied, only changing the mask, as previously explained in Section 2.1.3 and expressed with Equations (4)–(7). The inference system is a type-1 fuzzy Mamdani FIS with two inputs, one output, and three fuzzy rules. The general methodology is explained as follows, emphasizing that the numerical results used as an example are calculated with fuzzy Sobel.

1. Read the input image. First, we need to read the input image; in this case, the database is defined in Section 2.2.1 and illustrated in Figure 4;
2. Obtain the inputs for the inference system. The type-1 inference system considers two inputs; these are the gradients calculated with Equations (8) and (9). The inputs are labeled as $D_h$ and $D_v$, which represent the gradients in the horizontal and vertical positions, respectively. The two inputs stand for Gaussians membership functions expressed in Equation (3). The $D_h$ input is granulated in three MFs with the linguistic variables: “Low $D_h$”, “Middle $D_h$”, and “High $D_h$”; the $D_v$ input also has three MFs defined as “Low $D_v$”, “Middle $D_v$”, and “High $D_v$”. The parameters are determined according to the gradient values of each image, i.e., considering that the input image is Figure 5, we obtain lower with Equation (14), high using Equation (15), middle with Equation (16), and $\sigma$ values using Equation (17) for the gradients $D_h$ and $D_v$. 
tical positions, respectively. The two inputs stand for Gaussians membership functions expressed in Equation (3). The $D_h$ input is granulated in three MFs with the linguistic variables: "LowDh", "MiddleDh", and "HighDh"; the $D_v$ input also has three MFs defined as "LowDv", "MiddleDv", and "HighDv". The parameters are determined according to the gradient values of each image, i.e., considering that the input image is Figure 5, we obtain low with Equation (14), high using Equation (15), middle with Equation (16), and $\sigma$ values using Equation (17) for the gradients $D_h$ and $D_v$.

\begin{align*}
\text{lowDh} &= \min(D_h), \text{lowDv} = \min(D_v) \\
\text{highDh} &= \max(D_h), \text{highDv} = \max(D_v) \\
\text{MiddleDh} &= (\text{lowDh} + \text{highDh})/4, \text{MiddleDv} = (\text{lowDv} + \text{highDv})/4 \\
\sigma_{Dh} &= \text{highDh} / 4, \sigma_{Dv} = \text{highDv} / 4
\end{align*}

(14) (15) (16) (17)

Considering the gradients in Figure 6, we obtain the values $\text{lowDh} = 0$, $\text{MiddleDh} = 219.50$, $\text{highDh} = 878$, and $\sigma_{Dh} = 219.50$ for the input $D_h$. The parameters for the Gaussian MFs are expressed in Equations (18)–(20) and illustrated in Figure 6a.

\begin{align*}
\mu_{\text{LowDh}}(x) &= \exp\left[-\frac{1}{2}(x - 0)^2\right] \\
\mu_{\text{MiddleDh}}(x) &= \exp\left[-\frac{1}{2}(x - 219.50)^2\right] \\
\mu_{\text{HighDh}}(x) &= \exp\left[-\frac{1}{2}(x - 878)^2\right]
\end{align*}

(18) (19) (20)

Figure 4. Sample of the guitar database.

Sensors 2022, 22, x FOR PEER REVIEW 9 of 24
The Gaussian MFs for the Dv input are expressed in Equations (21)–(23) and illustrated in Figure 6b, with the values \( \text{lowDh} = 0, \text{MiddleDv} = 208.50, \text{highDv} = 834, \) and \( \sigma\text{Dv} = 208.50. \)

\[
\mu_{\text{LowDv}}(x) = \exp \left[ -\frac{1}{2} \left( \frac{x - 0}{208.50} \right)^2 \right] \tag{21}
\]

\[
\mu_{\text{MiddleDv}}(x) = \exp \left[ -\frac{1}{2} \left( \frac{x - 208.50}{208.50} \right)^2 \right] \tag{22}
\]

\[
\mu_{\text{HighDv}}(x) = \exp \left[ -\frac{1}{2} \left( \frac{x - 834}{208.50} \right)^2 \right] \tag{23}
\]

Figure 5. Input guitar image.

Figure 6. (a) Dh input MF; (b) Dv input MF.

1. Obtain the output. The T1FIS has one output labeled as Edges; this is divided in two linguistic values “Background” and “Edge”. In this study case, the output (Edges) is normalized in a range between −5 and 4.5. The center value for the Background MF is \( c_{\text{Background}} = -5 \) and for the Edge MF is defined as \( c_{\text{Edge}} = 4.5. \) The \( \sigma \) value for both MFs is calculated with Equation (24);

\[
\sigma_{\text{output}} = \text{abs}(c_{\text{Background}} - c_{\text{Edge}})/2
\]

The parameters for the output MF (Edges) are expressed in Equations (25) and (26); considering the values of \( c_{\text{Background}} = -5, c_{\text{Edge}} = 4.5, \) and \( \sigma_{\text{output}} = 4.75. \)

\[
\mu_{\text{Background}}(x) = \exp \left[ -\frac{1}{2} \left( \frac{x - (-5)}{4.75} \right)^2 \right] \tag{25}
\]
\[ \mu_{\text{Edge}}(x) = \exp \left[ -\frac{1}{2} \left( \frac{x - 4.5}{4.75} \right)^2 \right] \]  

(26)

2. Mamdani Fuzzy rules. The knowledge base is represented with three fuzzy rules (see Table 2); these are obtained based on expert knowledge. The fuzzy Sobel edge-detection output is illustrated in Figure 7a.

![Image](a) ![Image](b)

**Figure 7.** (a) Fuzzy Sobel edge-detection output. (b) Fuzzy MG output.

Algorithm 1 summarizes the process for calculating Prewitt and Sobel fuzzy edge-detection.

**Algorithm 1.** Fuzzy edge-detection using Prewitt and Sobel operators.

1. Select the operator to calculate the gradients (Prewitt or Sobel).
2. Case Prewitt:
3. Use kernels \( \text{Prewitt}_x \) (Equation (4)) and \( \text{Prewitt}_y \) (Equation (5))
4. Case Sobel:
5. Use kernels \( \text{Sobel}_x \) 9 (Equation (6)) and \( \text{Sobel}_y \) (Equation (7))
6. Read input image \( f \)
7. Obtain image dimensions (\( \text{rows} \) and \( \text{columns} \))
8. \( [\text{rows}, \text{columns}] = \text{size}(f) \)
9. Calculate the classical gradient corresponding to the fuzzy filter desired
10. \([\text{Dx}, \text{Dy}] = \text{zeros}([\text{rows}, \text{columns}]) \) // Generate a zero matrix with the same dimensions as \( f \) to capture the gradient
11. for \( i = 0 \) to \( \text{rows} \):
12. for \( j = 0 \) to \( \text{columns} \):
13. \( \text{Dx}[i,j] = \text{sum}(\text{kernel}_x \ast f[i:i+3, j:j+3]) \) // Equation (8)
14. \( \text{Dy}[i,j] = \text{sum}(\text{kernel}_y \ast f[i:i+3, j:j+3]) \) // Equation (9)
15. end for
16. end for
17. end for
18. Generate the required fuzzy controller using the fuzzy rules from Tables 2 and 3.
19. Fuzzify the two input gradients \( \text{Dx} \) and \( \text{Dy} \), using Gaussian MFs (Equations (18)–(23)).
20. Infer the edges output with the selected controller
21. Defuzzify the output (edges) of the controller.

**Table 2.** Fuzzy rules base for fuzzy MG edge-detector.

| G1   | G2   | G3   | G4   | Output          | Operator |
|------|------|------|------|-----------------|----------|
| HighG1 | HighG2 | HighG3 | HighG4 | Edge            | or       |
| MiddleG1 | MiddleG2 | MiddleG3 | MiddleG4 | Edge            | or       |
| LowG1  | LowG2  | LowG3  | LowG4  | Background      | and      |
Table 3. Knowledge base with three fuzzy rules for Sobel/Prewitt fuzzy edge-detector.

| Inputs | Output | Operator |
|--------|--------|----------|
| Dh     | Dh     | edge     | or      |
| Dv     | Dv     | edge     | or      |
| Edges  | Background and |

Fuzzy Morphological Gradient Edge-Detection

This edge-detector combines the T1 fuzzy set theory and the morphological gradient technique (explained in Section 2.1.3). A fuzzy Mamdani system is implemented that consists of four inputs, one output, and three fuzzy rules; the process is calculated with the following steps.

1. Obtain the four image gradients. After reading the input image, Equations (11) and (12) are used to calculate the image gradients in all directions ($G_1$, $G_2$, $G_3$, and $G_4$). Each gradient represents an input in the fuzzy inference system;

2. Define the input MFs and their respective linguistic variables. T1FIS consists of four inputs ($G_1$, $G_2$, $G_3$, and $G_4$); each one is separated into three Gaussian membership functions with the linguistic variables “low”, “medium”, and “high”. The parameters for each input MF are adapted depending on the image gray tones; therefore, to calculate the parameters, we obtain the $Low$, $Middle$, and $High$ values for each gradient or input using Equations (27)–(30), and these values are used to calculate the $\sigma$ (Equation (31));

$$lowG_1 = \min(G_1), lowG_2 = \min(G_2), lowG_3 = \min(G_3), lowG_4 = \min(G_4)$$ (27)

$$highG_1 = \max(G_1), highG_2 = \max(G_2), highG_3 = \max(G_3), highG_4 = \max(G_4)$$ (28)

$$mediumG_1 = (lowG_1 + highG_1)/2, mediumG_2 = (lowG_2 + highG_2)/2$$ (29)

$$mediumG_3 = (lowG_3 + highG_3)/2, mediumG_4 = (lowG_4 + highG_4)/2$$ (30)

$$\sigma G_1 = \frac{1}{4\sigma G_1}, \sigma G_2 = \frac{1}{4\sigma G_2}, \sigma G_3 = \frac{1}{4\sigma G_3}, \sigma G_4 = \frac{1}{4\sigma G_4}$$ (31)

After calculating the four gradients in Figure 6, we obtain the values of $lowG_1 = 0, MiddleG_1 = 219.50, highG_1 = 878, and \sigma G_1 = 219.50$ for the input $G_1; lowG_2 = 0, MiddleG_2 = 219.50, highG_2 = 878, and \sigma G_2 = 219.50$ for the input $G_2; lowG_3 = 0, MiddleG_3 = 219.50, highG_3 = 878, and \sigma G_3 = 219.50$ for the input $G_3; lowG_4 = 0, MiddleG_4 = 219.50, highG_4 = 878, and \sigma G_4 = 219.50$ for the input $G_4$. According to these parameters, the Gaussian MFs are denoted by Equations (32)–(43), and Figure 8 illustrates the inputs $G_1$ and $G_2$.

$$\mu LowG_1(x) = \exp \left( -\frac{1}{2} \left( \frac{x-0}{50.5890} \right)^2 \right)$$ (32)

$$\mu MiddleG_1(x) = \exp \left( -\frac{1}{2} \left( \frac{x-101.1781}{50.5890} \right)^2 \right)$$ (33)

$$\mu HighG_1(x) = \exp \left( -\frac{1}{2} \left( \frac{x-202.3561}{50.5890} \right)^2 \right)$$ (34)

$$\mu LowG_2(x) = \exp \left( -\frac{1}{2} \left( \frac{x-0}{46.1912} \right)^2 \right)$$ (35)

$$\mu MiddleG_2(x) = \exp \left( -\frac{1}{2} \left( \frac{x-92.3824}{46.1912} \right)^2 \right)$$ (36)
\[\mu_{\text{HighG}2}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 184.7647}{219.50}\right)^2\right)\] (37)

\[\mu_{\text{LowG}3}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 0}{58.1893}\right)^2\right)\] (38)

\[\mu_{\text{MiddleG}3}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 116.3787}{58.1893}\right)^2\right)\] (39)

\[\mu_{\text{HighG}3}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 232.7574}{58.1893}\right)^2\right)\] (40)

\[\mu_{\text{LowG}4}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 0}{44.5884}\right)^2\right)\] (41)

\[\mu_{\text{MiddleG}4}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 89.1768}{44.5884}\right)^2\right)\] (42)

\[\mu_{\text{HighG}4}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 178.3536}{44.5884}\right)^2\right)\] (43)

3. Calculate the output. T1FIS consists of one output (Edges), granulated with two linguistic values "Background" and "Edge". In the same manner as in the fuzzy Sobel and fuzzy Prewitt approaches, the output edges matrix is normalized in a range between −5 and 4.5; the center value for the Background MF is \(c_{\text{Background}} = -5\) and the Edge MF is \(c_{\text{Edge}} = 4.5\). The \(\sigma\) value for both MFs is calculated with Equation (44);

\[\sigma_{\text{output}} = \text{abs}(c_{\text{Background}} - c_{\text{Edge}}) / 2\] (44)

The parameters for the output MF (Edges) are expressed in Equations (45) and (46) and illustrated in Figure 9, considering the values of \(c_{\text{Background}} = -5\), \(c_{\text{Edge}} = 4.5\), and \(\sigma_{\text{output}} = 4.75\).

\[\mu_{\text{Background}}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - (-5)}{4.75}\right)^2\right)\] (45)

\[\mu_{\text{Edge}}(x) = \exp\left(-\frac{1}{2} \left(\frac{x - 4.5}{4.75}\right)^2\right)\] (46)

4. Mamdani Fuzzy rules. The knowledge base is represented with three fuzzy rules (Table 2). The fuzzy MG edge-detection output is illustrated in Figure 7b.

2.2.3. CNN Training and Validation

For this research, we experimented with three architectures: two proposed models of convolutional neural networks and a pre-trained VGG16 model [49]. The proposed models were crafted with the ideology of achieving high levels of accuracy without consuming large amounts of computing resources, such as the VGG16 or Inception models [50], which are very large models trained for multiple categories. In this case, the model CNN-I (architecture overview in Table 4) consists of four convolution layers that include the rectified linear unit activation (ReLu) and maximum pooling in between each convolution layer, a dropout of 45%, two dense layers, and the output layer. The CNN-II model (architecture overview in Table 5) is a variant of the first model where the neurons in the convolutional layers are increased to extract as much information as possible from the filter; as well as CNN-I, each of these convolution layers includes a ReLu activation, and each convolution is followed by a maximum pooling layer with the same filter size. To compensate for overfitting the number of neurons used, the dropout layer is increased.
to turn off 75% of the neurons before passing to a single, but significantly larger, fully connected layer and finally going to the output.

\[
\mu_{E_{\text{edge}}}(x) = e^{x \cdot p_{\text{edge}} - 1} \left( x - 4.5 \right)^{4.75/6}
\] (46)

4. Mamdani Fuzzy rules. The knowledge base is represented with three fuzzy rules (Table 2). The fuzzy MG edge-detection output is illustrated in Figure 7b.

Figure 8. Inputs MFs for the Fuzzy MG.

Figure 9. Output MF for the Fuzzy MG.

Table 4. Proposed CNN-I.

| Layer/Type               | Filters/Neurons | Filter Size |
|--------------------------|-----------------|-------------|
| 0/Input Convolution + ReLu| 40              | 3 × 3       |
| 1/Max Pooling            | –               | 2 × 2       |
| 2/Convolution + ReLu     | 40              | 3 × 3       |
| 3/Max Pooling            | –               | 2 × 2       |
| 4/Convolution+ ReLu      | 60              | 3 × 3       |
| 5/Max Pooling            | –               | 2 × 2       |
| 6/Convolution + ReLu     | 60              | 3 × 3       |
| 7/Max Pooling            | –               | 2 × 2       |
| 8/Dropout                | –               | 0.45        |
| 9/Flatten                | –               | –           |
| 10/Dense                 | 300             | –           |
| 11/Dense                 | 250             | –           |
| 12/Output                | 6               | –           |
Table 5. Proposed CNN-II.

| Layer/Type                      | Filters/Neurons | Filter Size |
|---------------------------------|-----------------|-------------|
| 0/Input Convolution + ReLu      | 64              | 3 × 3       |
| 1/Max Pooling                   | –               | 2 × 2       |
| 2/Convolution + ReLu            | 64              | 3 × 3       |
| 3/Max Pooling                   | –               | 2 × 2       |
| 4/Convolution + ReLu            | 128             | 3 × 3       |
| 5/Max Pooling                   | –               | 2 × 2       |
| 6/Convolution + ReLu            | 128             | 3 × 3       |
| 7/Max Pooling                   | –               | 2 × 2       |
| 8/Dropout                       | –               | 0.75        |
| 9/Flatten                       | –               | –           |
| 10/Dense                        | 512             | –           |
| 11/Output                       | 6               | –           |

The hyperparameters used to train the models were determined by experimentation. After many trials, the best results were obtained using the following parameters:

- **Epochs**: 30;
- **Loss function**: Sparse categorical cross entropy;
- **Training algorithm**: Adam (default settings in Keras API);
  - **Epsilon**: $1 \times 10^{-7}$;
  - **Beta 1**: 0.9;
  - **Beta 2**: 0.999;
  - **Learning rate**: 0.001;
- **Batch size**: 60.

In the case of the pre-trained model, we used a VGG16 from the TensorFlow library, using the weights from the ImageNet dataset [51]. The weights were updated by performing transfer learning. This was done by removing the last (output) layer of the model and freezing the training parameters of the other layers. A new single classification layer was added with six neurons (one for each class) and we retrained the model with the following parameters:

- **Epochs**: 15;
- **Loss function**: Sparse categorical cross entropy;
- **Training algorithm**: Adam (default settings in Keras API);
  - **Epsilon**: $1 \times 10^{-7}$;
  - **Beta 1**: 0.9;
  - **Beta 2**: 0.999;
  - **Learning rate**: 0.001;
- **Batch size**: 80.

For the training of these models, we performed K-fold cross-validation; in this process, division of the dataset was performed to create multiple training and test sets denominated as folds. This process was performed to evaluate the model with the entire dataset. Fuzzy preprocessing showed an overall performance improvement over the non-preprocessed images (grayscale and RGB).

3. Results

To compare the efficiency of the proposed models, we implemented them in Python, using the TensorFlow framework with the Keras API. Summaries of the results of the 30 experiments performed for each preprocessing approach for CNN-I and CNN-II are shown in Tables 6 and 7, respectively, and the results for the VGG16 model are displayed in Table 8. Fuzzy preprocessing showed an overall performance improvement against the grayscale images and the classic edge-detection methodologies. The experiments
were performed on a laptop with the following specifications: CPU Intel Core i7-11800H, 32 GB DDR4 RAM at 3200 MHz, and an Nvidia RTX 3080 Laptop GPU with 16 GB of video memory.

Table 6. Results of the CNN-I model.

| Preprocessing | Min.    | Max.    | Average | Standard Deviation |
|---------------|---------|---------|---------|-------------------|
| RGB           | 0.6296  | 0.7127  | 0.6687  | 0.0141            |
| Grayscale     | 0.6135  | 0.7037  | 0.6601  | 0.0199            |
| MG            | 0.4870  | 0.5759  | 0.5307  | 0.0177            |
| Prewitt       | 0.6305  | 0.7129  | 0.6763  | 0.0163            |
| Sobel         | 0.5894  | 0.6956  | 0.6425  | 0.0282            |
| Fuzzy MG      | 0.6315  | 0.7167  | 0.6708  | 0.0186            |
| Fuzzy Prewitt | 0.6380  | 0.7176  | 0.6775  | 0.0151            |
| Fuzzy Sobel   | 0.6380  | 0.7176  | 0.6731  | 0.0196            |

Table 7. Results of the CNN-II model.

| Preprocessing | Min.    | Max.    | Average | Standard Deviation |
|---------------|---------|---------|---------|-------------------|
| RGB           | 0.6796  | 0.7491  | 0.7121  | 0.0137            |
| Grayscale     | 0.6654  | 0.7463  | 0.7031  | 0.0176            |
| MG            | 0.5102  | 0.5991  | 0.5597  | 0.0177            |
| Prewitt       | 0.6675  | 0.7481  | 0.7130  | 0.0139            |
| Sobel         | 0.6712  | 0.6898  | 0.6789  | 0.01794           |
| Fuzzy MG      | 0.6657  | 0.7444  | 0.7041  | 0.0153            |
| Fuzzy Prewitt | 0.6824  | 0.7481  | 0.7157  | 0.0145            |
| Fuzzy Sobel   | 0.1667  | 0.7518  | 0.7030  | 0.0479            |

Table 8. Results of the VGG16.

| Preprocessing | Min.    | Max.    | Average | Standard Deviation |
|---------------|---------|---------|---------|-------------------|
| RGB           | 0.6398  | 0.7356  | 0.6908  | 0.0203            |

To evaluate the performance between the fuzzy preprocessing techniques and the color images, we calculated a ROC curve to have a visual representation of the performance of each class. Curves were calculated for both proposed models, as shown in Figure 10 for CNN-I and Figure 11 for CNN-II, each time with the best instance of the models trained.

To expand the results of the ROC graph where the RGB images presented a slightly lower accuracy when compared to some of the fuzzy detectors, we calculated the average training time for each model. Figure 12 contains the average training time per model in seconds for CNN-I, and Figure 13 presents the averages for CNN-II.
To evaluate the performance between the fuzzy preprocessing techniques and the color images, we calculated a ROC curve to have a visual representation of the performance of each class. Curves were calculated for both proposed models, as shown in Figure 10 for CNN-I and Figure 11 for CNN-II, each time with the best instance of the models trained.

Figure 10. ROC curve for CNN-I.

Figure 11. ROC curve for CNN-II.
The tests for each preprocessing methodology were made independent for the model. The Z-test was a right-tailed test; the parameters used for the tests were the following:

- Right-tailed test;
- $\alpha = 0.05$ (95% confidence level, rejection zone at $z_c = 1.96$);
- $n = 30$;
- $H_0$: Fuzzy preprocessing approach ($\mu_1$) offers less or equal accuracy than the raw images ($\mu_2$). $H_0$: $\mu_1 \leq \mu_2$;
- $H_a$: Fuzzy preprocessing approach ($\mu_1$) offers more accuracy than the raw images ($\mu_2$). $H_a$: $\mu_1 > \mu_2$ (affirmation).

The results of the null hypothesis test for CNN-I are shown in Table 9. The $\bar{x}_1$ and $\sigma_1$ variables represent the mean and standard deviation, respectively, for the fuzzy preprocessing approach; $\bar{x}_2$ and $\sigma_2$ represent the mean and standard deviation, respectively, for raw images and grayscale.
Table 9. Hypothesis test results for CNN-I model.

| Preprocessing       | $\bar{x}_1$ | $\sigma_1$ | $\bar{x}_2$ | $\sigma_2$ | Z-Score |
|---------------------|------------|------------|------------|------------|---------|
| Fuzzy Sobel vs. RGB | 0.6708     | 0.0186     | 0.6687     | 0.0141     | 1.4347  |
| Fuzzy Prewitt vs. RGB | 0.6775    | 0.0151     | 0.6687     | 0.0141     | 2.3336  |
| Fuzzy MG vs. RGB    | 0.6751     | 0.0196     | 0.6687     | 0.0141     | 0.4892  |
| Fuzzy Sobel vs. grayscale | 0.6708  | 0.0186     | 0.6601     | 0.0199     | 2.9244  |
| Fuzzy Prewitt vs. grayscale | 0.6775 | 0.0151     | 0.6601     | 0.0199     | 3.8144  |
| Fuzzy MG vs. grayscale | 0.6751   | 0.0196     | 0.6601     | 0.0199     | 2.1474  |

The results for the Z-test for the CNN-II model are then shown in Table 10.

Table 10. Hypothesis test results for CNN-II model.

| Preprocessing       | $\bar{x}_1$ | $\sigma_1$ | $\bar{x}_2$ | $\sigma_2$ | Z-Score |
|---------------------|------------|------------|------------|------------|---------|
| Fuzzy Sobel vs. color | 0.7041    | 0.0153     | 0.7121     | 0.0137     | -0.9961 |
| Fuzzy Prewitt vs. color | 0.7157   | 0.0145     | 0.7121     | 0.0137     | 1.0065  |
| Fuzzy MG vs. color    | 0.7030     | 0.0479     | 0.7121     | 0.0137     | -2.1186 |
| Fuzzy Sobel vs. grayscale | 0.7041   | 0.0153     | 0.7031     | 0.0176     | -0.006  |
| Fuzzy Prewitt vs. grayscale | 0.7157  | 0.0145     | 0.7031     | 0.0176     | 3.0494  |
| Fuzzy MG vs. grayscale | 0.7030    | 0.0479     | 0.7031     | 0.0176     | 0.2462  |

4. Discussion

As demonstrated in the literature review, in addition to the results obtained, implementing fuzzy image preprocessing techniques before feeding the images into a convolutional neural network has proven to be beneficial in most instances for improving the accuracy of the model. As shown in Tables 6 and 7, the maximum accuracy was obtained with fuzzy Sobel and fuzzy Prewitt for CNN-I, with a value of 71.76% in both cases after performing the 30 experiments; in the case of the CNN-II, fuzzy Sobel achieved the best model performance, obtaining the maximum accuracy of 75.18%. These are competitive results against RGB images, where the latter do not require preprocessing but need more computational resources and time to train the model.

When reviewing the classical edge-detection methodologies for CNN-I, we can note that Sobel and MG preprocessing tended to decrease the accuracy, with an average of 53.07% for MG and 64.72% for Sobel. With Prewitt, we obtained the best average for the model with 67.63%, even better than the RGB images, which gave a result of 66.87%. The average for the fuzzy preprocessing offered a slight improvement of almost 1% when compared to the RGB images in all instances. The results obtained with CNN-II also followed a tendency where the classic Sobel and MG offered a decrease in the average of almost 15% for MG and 4% for Sobel. We also noted comparable averages with the fuzzy approaches and with Prewitt preprocessing; all four options were comparable to RGB images with a delta of ±1%.

In the case of CNN-II, even though the model had an overall accuracy improvement, when compared to CNN-I, we can note a similar pattern to CNN-I where classical edge-detection methodologies suffered an accuracy loss compared to grayscale images, though applying fuzzy filters allowed us to surpass the grayscale approach. In this case, the performance loss with MG edge-detection was more significant, with a 14.34% lower accuracy. In the case of the Sobel edge-detection, we had a 2.41% accuracy loss on average. Fuzzy Prewitt reported a 1.08% accuracy improvement when compared to the grayscale preprocessing, and a 0.18% accuracy improvement when compared to the color images.

The results obtained with the pre-trained VGG16 model, denoted in Table 8, were not as impressive as those for the specialized model without preprocessing, especially when compared to CNN-II with fuzzy Sobel, which gave over a 1.9% accuracy improvement with the best models. This alternative could represent a decent out-of-the-box implementation,
without taking into consideration the training times needed, because the model utilizes $224 \times 224$ px RGB images instead of the $150 \times 150$ px grayscale used for fuzzy Sobel.

The results can be validated with the ROC curves shown in Figures 10 and 11, which show the sensitivity and specificity of the models. We can observe that each class in every model has a similar performance, where the models are performing a decent separation of the classes. The models present a similar level of area under the curve (AUC) due to the relatively small differences between them; we found no more than a 2% difference between preprocessing methodologies and no more than a 5% difference between the best CNN-I models compared to the best of CNN-II models.

The ROC curves shown in Figures 10 and 11 demonstrate similar behavior in terms of accuracy between the fuzzy preprocessing and the color (RGB) images. To evaluate the effectiveness of the proposed methodologies in both architectures, we performed a time analysis by training 30 independent models, to evaluate the average training time, and we calculated the average preprocessing time for the images (Figures 12 and 13). With CNN-I, we noted a reduction by an average of 23% in the training time with the fuzzy preprocessing. In the case of CNN-II, we had a reduction in time of 18.5% with the more complex architecture. In both instances, we managed to achieve similar accuracy rates with a significant reduction in training time. These reductions can be significant in similar applications in which the models need to be trained with more information.

The models trained with a preprocessing step showed a clear advantage in training time when compared to the color images, due to the utilization of a single layer, instead of the three layers used in RGB images. The trade-off between the optimization in training times and doubling the preprocessing step time is directly reflected in GPU usage, which normally represents the highest upfront cost when compared to CPU usage, in the development of these types of systems.

The results of the statistical analysis performed, demonstrated in Tables 9 and 10, show that for CNN-I, the comparison between the utilization of fuzzy preprocessing against RGB images revealed a significant difference with the fuzzy Prewitt application, with a Z-score of 2.3. In the other two instances, the fuzzy filters presented better accuracy, but the threshold was not reached. On the other hand, when comparing the fuzzy filters against the grayscale images, we can note that all the Z-scores surpassed the 1.96 threshold, therefore, providing enough evidence to accept our alternative hypothesis, making the fuzzy preprocessing step a compelling alternative to improve accuracy. The results obtained for CNN-II demonstrated that there is sufficient evidence to validate the usage of a fuzzy filter (Prewitt) when compared to the grayscale model. The main support for this was the lack of margin between the results, where the margins of improvement in precision were not as tangible as for CNN-I, which had less capacity to extract characteristics from which it benefitted on multiple occasions with the use of fuzzy edge-detectors.

5. Conclusions and Future Work

The experimentation and statistical analysis showed that implementing fuzzy edge-detection for images before feeding them into the convolutional layers in a DNN (in some instances) can significantly improve the accuracy of the trained models, while reducing training times, especially in model CNN-I, where less filters are used for feature extraction. On other hand, model CNN-II, with more filters per layer, demonstrated improved accuracy, but no significant evidence was found to validate the usage of preprocessing layers.

When investigating the pre-trained VGG16 model, a 2% reduction in accuracy was found compared to CNN-2 with the RGB images, making it a poor alternative, especially considering the model size and training times.

We believe that the main limiting factors that affected our results were the complexity of our training information, with vast differences between images in the same category of the dataset, as well as similarities between the classes. In the future, we would like to improve the dataset and develop a more curated version that does not include the huge variation in the current version, and we would like to implement the proposed
preprocessing methodology and model to benchmark datasets, to compare the efficiency against state-of-the-art models.

**Author Contributions:** Conceptualization, C.I.G.; methodology, C.I.G. and G.E.M.; software, C.T.; validation, C.I.G. and G.E.M.; formal analysis, C.T.; investigation, C.T.; data curation, C.T.; writing—original draft preparation, C.T.; writing—review and editing, C.T., C.I.G. and G.E.M.; supervision, G.E.M.; project administration, C.I.G.; funding acquisition, C.I.G.; formal analysis, C.I.G.; conceptualization and writing—review and editing, G.E.M. and C.I.G.; methodology, G.E.M.; investigation, software, data curation and writing—original draft preparation, C.I.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by CONACYT, grant number 1106407, and Tijuana Institute of Technology, project number IT18A85.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** We thank Tijuana Institute of Technology/TECNM and CONACYT for support with the finances that enabled us to sustain this investigation.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Zhang, Y.; Dong, Z.; Chen, X.; Jia, W.; Du, S.; Muhammad, K.; Wang, S. Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimed. Tools Appl.* 2017, 78, 3613–3632. [CrossRef]  
2. Lin, C.H.; Yu, C.C.; Wang, T.Y.; Chen, T.Y. Classification of the tree for aerial image using a deep convolution neural network and visual feature clustering. *J. Supercomput.* 2020, 76, 2503–2517. [CrossRef]  
3. Sebti, R.; Zroug, S.; Kahloul, L.; Benharzallah, S. A Deep Learning Approach for the Diabetic Retinopathy Detection. In *Innovations in Smart Cities Applications Volume 5. SCA 2021. Lecture Notes in Networks and Systems*; Ben Ahmed, M., Boudhir, A.A., Karas, I.R., Jain, V., Mellouli, S., Eds.; Springer: Cham, Switzerland, 2022; Volume 393, pp. 459–469.  
4. Castro Rabelo Profeta, R.; Schuller, G. Feature-Based Classification of Electric Guitar Types. In *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2019. Communications in Computer and Information Science*; Cellier, P., Driessens, K., Eds.; Springer: Cham, Switzerland, 2020; Volume 1168, pp. 478–484.  
5. Banerjee, A.; Ghosh, A.; Palit, S.; Ballester, M.A.F. A Novel Approach to String Instrument Recognition. In *Image and Signal Processing. ICISP 2018. Lecture Notes in Computer Science*; Mansouri, A., El Moataz, A., Nouboud, F., Mannmass, D., Eds.; Springer: Cham, Switzerland, 2018; Volume 10884, pp. 165–175.  
6. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 1986, 8, 679–698. [CrossRef] [PubMed]  
7. Hu, L.; Cheng, H.D.; Zhang, M. A high performance edge detector based on fuzzy inference rules. *Inf. Sci.* 2007, 177, 4768–4784. [CrossRef]  
8. Ontiveros-Robles, E.; Vázquez, J.G.; Castro, J.R.; Castillo, O. A FPGA-Based Hardware Architecture Approach for Real-Time Fuzzy Edge Detection. In *Nature-Inspired Design of Hybrid Intelligent Systems. Studies in Computational Intelligence*; Melin, P., Castillo, O., Kacprzyk, J., Eds.; Springer: Cham, Switzerland, 2017; Volume 667, pp. 519–540.  
9. Mendoza, O.; Melin, P.; Licea, G. Interval type-2 fuzzy logic for edges detection in digital images. *Int. J. Intell. Syst.* 2009, 24, 1115–1133. [CrossRef]  
10. Melin, P.; Mendoza, O.; Castillo, O. A New Method for Edge Detection in Image Processing Using Interval Type-2 Fuzzy Logic. In *Proceedings of the IEEE International Conference on Granular Computing (GRC 2007)*, Fremont, CA, USA, 2 November 2007.  
11. Biswas, R.; Sil, J. An Improved Canny Edge Detection Algorithm Based on Type-2 Fuzzy Sets. *Procedia Technol.* 2012, 4, 820–824. [CrossRef]  
12. Melin, P.; Mendoza, O.; Castillo, O. An Improved method for edge detection based on interval type-2 fuzzy logic. *Expert Syst. Appl.* 2010, 37, 8527–8535. [CrossRef]  
13. Gonzalez, C.I.; Melin, P.; Castro, J.R.; Mendoza, O.; Castillo, O. General Type-2 Fuzzy Edge Detection in the Preprocessing of a Face Recognition System. In *Nature-Inspired Design of Hybrid Intelligent Systems. Studies in Computational Intelligence*; Melin, P., Castillo, O., Kacprzyk, J., Eds.; Springer: Cham, Switzerland, 2017; Volume 667, pp. 3–18.  
14. Gonzalez, C.I.; Castro, J.R.; Mendoza, O.; Melin, P. General type-2 fuzzy edge detector applied on face recognition system using neural networks. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Vancouver, BC, Canada, 24 July 2016.
42. Kirsch, R. Computer determination of the constituent structure of biological images. *Comput. Biomed. Res.* 1971, 4, 315–328. [CrossRef]
43. Sobel, I. Camera Models and Perception. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1970.
44. Prewitt, J.M.S. Object enhancement and extraction. *Pict. Anal. Psychopictorics* 1970, 75–149.
45. Martínez, G.E.; Gonzalez, C.I.; Mendoza, O.; Melin, P. General Type-2 Fuzzy Sugeno Integral for Edge Detection. *J. Imaging* 2019, 5, 71. [CrossRef]
46. Gibson. Available online: https://www.gibson.com/en-US/ (accessed on 3 April 2021).
47. Fender. Available online: https://www.fender.com/en/start (accessed on 3 April 2021).
48. Ibanez. Available online: https://www.ibanez.com/na/ (accessed on 3 April 2021).
49. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
50. Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7 June 2015.
51. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* 2015, 115, 221–252. [CrossRef]