mathematics

MDPI

Article

Investigating Effective Geometric Transformation for Image Augmentation to Improve Static Hand Gestures with a Pre-Trained Convolutional Neural Network

Baiti-Ahmad Awaluddin [1,2], Chun-Tang Chao [1] and Juing-Shian Chiou [1,*]

[1] Department of Electrical Engineering, Southern Taiwan University of Science and Technology, 1, Nan-Tai St., Yongkang District, Tainan 71005, Taiwan; da82b207@stust.edu.tw (B.-A.A.); tang@stust.edu.tw (C.-T.C.)
[2] Department of Electronics Engineering Education, Universitas Negeri Yogyakarta, Yogyakarta 55281, Indonesia
* Correspondence: jschiou@stust.edu.tw; Tel.: +886-916-221-152; Fax: +886-6-3010-069

Abstract: Hand gesture recognition (HGR) is a challenging and fascinating research topic in computer vision with numerous daily life applications. In HGR, computers aim to identify and classify hand gestures. The limited diversity of the dataset used in HGR is due to the limited number of hand gesture demonstrators, acquisition environments, and hand pose variations despite previous efforts. Geometric image augmentations are commonly used to address these limitations. These augmentations include scaling, translation, rotation, flipping, and image shearing. However, research has yet to focus on identifying the best geometric transformations for augmenting the HGR dataset. This study employed three commonly utilized pre-trained models for image classification tasks, namely ResNet50, MobileNetV2, and InceptionV3. The system's performance was evaluated on five static HGR datasets: DLSI, HG14, ArabicASL, MU HandImages ASL, and Sebastian Marcell. The experimental results demonstrate that many geometric transformations are unnecessary for HGR image augmentation. Image shearing and horizontal flipping are the most influential transformations for augmenting the HGR dataset and achieving better classification performance. Moreover, ResNet50 outperforms MobileNetV2 and InceptionV3 for static HGR.

Keywords: hand gesture recognition; image augmentation; geometric transformation; ResNet; MobileNet; inception; static datasets

MSC: 68T07

Citation: Awaluddin, B.-A.; Chao, C.-T.; Chiou, J.-S. Investigating Effective Geometric Transformation for Image Augmentation to Improve Static Hand Gestures with a Pre-Trained Convolutional Neural Network. *Mathematics* **2023**, *11*, 4783. https://doi.org/10.3390/math11234783

Academic Editors: Adrian Sergiu Darabant, Diana-Laura Borza and Faheim Sufi

Received: 21 October 2023
Revised: 20 November 2023
Accepted: 24 November 2023
Published: 27 November 2023

1. Introduction

Interacting with computers using hand gestures can provide users with a natural and intuitive interface. As a result, much research has focused on developing more accurate and effective hand gesture recognition (HGR) methods and applying them in various contexts. Hand gestures are currently utilized in multiple applications, such as games [1,2], virtual and augmented reality [3–5], assisted living [6,7], and cognitive development evaluation [8]. In addition, hand gesture recognition has gained significant interest in several industries, including human–robot interaction in manufacturing [9–11] and autonomous vehicle control [12,13]. With the recent growth of HGR, there is an increasing demand for more advanced and robust methods to meet the requirements of various applications.

Despite the remarkable success of deep neural networks in HGR [14,15], there are still significant challenges in this research area. The complexity of hand gestures and differences in hand size are just two factors that can affect the performance of recognition algorithms. The training of deep neural networks with insufficient data can lead to overfitting or failure to learn a high-performance model. Various training schemes, including dropout layers and data augmentation techniques [16], have been proposed to address this challenge.

Data augmentation is a popular technique for increasing the size of a dataset and addressing the problem of insufficient data [17,18]. Various approaches to image-based data augmentation exist, encompassing geometric transformations, color manipulations, random occlusion, and methods grounded in deep learning, such as Generative Adversarial Networks (GANs) [18]. Among these, geometric transformations—comprising image scaling, rotation, translation, shearing, and flipping—stand out as one of the most prevalent approaches. These operations are crucial in expanding the sample pool for training deep neural networks, balancing dataset sizes, and enhancing overall efficiency [19].

Geometric transformations have been extensively applied in hand gesture recognition (HGR) studies to augment datasets effectively. For example, in Ref. [20], geometric augmentation demonstrated a noteworthy enhancement, improving CNN performance by up to 5%. Another study focusing on HGR, utilizing capsule networks, showcased improved results when combined with geometric augmentation involving rotation and translation operations [21]. Similarly, Ref. [22] used an adapted CNN and image translation (both vertically and horizontally) to augment original data, resulting in a notable 4% boost in classification accuracy. Moreover, Ref. [23] utilized random scaling and horizontal/vertical translation to increase the diversity of training data for HGR applications.

The integration of color transformations with geometric transformations has notably enhanced the performance of HGR systems. Color transformations involve histogram equalization, contrast or brightness enhancement, white balancing, sharpening, and blurring [24]. For instance, in Ref. [25], combining the shearing transformation with sigmoid and gamma correction augmented the original images, resulting in a 5% improvement in accuracy.

In their research, Taylor and Nicthe [26] highlighted the effectiveness of data augmentation methods in enhancing the classification task performance of CNNs. Specifically, their evaluation of various data augmentation schemes using a relatively simple CNN architecture showed that geometric augmentation methods outperformed photometric methods when training on a coarse-grained dataset, and these findings underscore the importance of augmenting coarse-grained training datasets using transformations that alter the geometry of images rather than focusing solely on lighting and color modifications.

GANs can be utilized for data augmentation [27,28] by training them to generate new synthetic data. GANs face challenges due to the potential for mode collapse, non-convergence, and oscillatory behavior [29,30]. There are differences between synthetic data, which are artificially created without using actual datasets, and the term "augmented data", which involves generating additional training data through modifications or transformations to the primary data. The primary objective of augmented data is to expand the diversity of the training set, prevent overfitting, and enhance the model's generalization capacity to previously unseen data. Data augmentation techniques can be applied to various data types, including images, text, and audio. On the other hand, augmented data provide significant advantages in improving the performance of deep learning models across various applications, such as object detection, image classification, image recognition, natural language understanding, semantic segmentation, and more [31]. This method has enhanced the efficiency and outcomes of deep learning models by generating new and diverse training examples for datasets. Moreover, the use of augmented data can also reduce operational costs related to the collection and labeling of data for deep learning models [28]. Deep learning models often require time-consuming and expensive operations for data collection and labeling.

Numerous studies demonstrate the superiority of geometric transformations over other methods. Furthermore, when aiming for generalization, a model capable of recognizing new datasets is essential. A CNN model requires sufficient depth, trained on large datasets, which demands substantial computational resources. Hence, an alternative is needed, and that comes in the form of using pre-existing models. These models have been trained beforehand with ample resources and can be applied to new datasets—a process known as transfer learning. There are numerous available CNN models that can

be utilized. In this study, three pre-trained CNN models are employed for static hand gesture recognition (HGR) tasks, namely ResNet50, MobileNetV2, and InceptionV3, chosen for their outstanding performance [32]. Therefore, this paper aims to address the following objectives:

1. Investigate whether using as many augmentations as possible on geometric transformations or focusing on a subset of the most effective geometric transformations yields better results in improving model performance.
2. Identify augmentation methods within geometric transformations that yield the highest accuracy rates in enhancing CNN performance, which is achieved through a systematic analysis. This involves a thorough examination of various geometric transformations, assessing their impact on the model's accuracy. The selection process is based on rigorous experimentation and quantitative evaluation, ensuring that the chosen augmentation methods contribute significantly to the improved performance of Convolutional Neural Networks (CNNs) in the context of static hand gesture recognition. The effectiveness is substantiated by comparative analyses and statistical measures, providing a robust foundation for the identified augmentation methods.
3. Compare the performance of three pre-trained models, ResNet50, MobileNetV2, and InceptionV3, in the classification of static hand gestures (HGRs). The evaluation of these three models is conducted to assess their ability to classify static hand gestures, providing crucial insights for further development in this field.
4. This research undertakes an evaluation of the accuracy of pre-trained neural networks for image classification. Section 2 describes research methodology. Section 3 describes the dataset, image augmentation, geometric transformations, CNN theory, and pre-trained neural networks (ResNet, MobileNet, and Inception). Section 4 presents the experimental setup, dataset preparation, and results using single and combined neural networks. Section 5 analyzes the performance of each pre-trained neural network and discusses the impact of image augmentation and geometric transformations on model accuracy. Additionally, we explore implications and future research opportunities in this area. Finally, Section 6 summarizes important findings, highlights our research's significance, and outlines future directions, including developing an image augmentation framework for static hand gesture recognition based on pre-trained ResNet50 models that combine multiple geometric transformations with color modifications.

2. Research Methodology

In this research, the steps to determine the best geometric transformation for image augmentation to improve recognition accuracy in hand gesture recognition (HGR) tasks and compare the performance of the pre-trained CNN models (ResNet50, MobileNetV2, and InceptionV3) are executed through a structured methodological approach. An illustration of the steps in the research methodology can be found in Figure 1. The researchers formulated the following research objectives:

1. Investigation of the necessity of employing geometric transformations for image augmentation in CNN-based HGR tasks;
2. Exploration of the optimal geometric transformation based on CNNs for image augmentation in HGR tasks;
3. Determining the most effective pre-trained CNN model (ResNet50, MobileNetV2, or Inceptionv3) for HGR tasks.

Firstly, diverse datasets, such as HG14, DLSI, MU HandImages ASL, Sebastian Marcel, and ArASL2018, were collected to encompass as many HGR contexts as possible. The next step involves the application of transfer learning to pre-trained models, namely ResNet50, MobileNetV2, and InceptionV3, to comprehend complex hand gesture features.

Evaluation is conducted in two main aspects, namely the effectiveness of image augmentation and the performance of pre-trained models, using accuracy metrics during transfer learning. Geometric transformations, such as scaling, rotation, translation, shearing, and flipping, are explored in the augmentation evaluation. At the same time, the

performance of pre-trained models is measured by comparing ResNet50, MobileNetV2, and InceptionV3 in classifying HGR datasets.

This research evaluates results and presents in-depth conclusions based on experimental findings. Overall, these methodological steps form a comprehensive framework to understand the role of image augmentation and CNN models in improving the accuracy of HGR systems. By implementation programming using Python 3.6.13, particularly with TensorFlow support, a solid technical foundation is provided, ensuring efficiency and optimal performance in conducting this experiment.

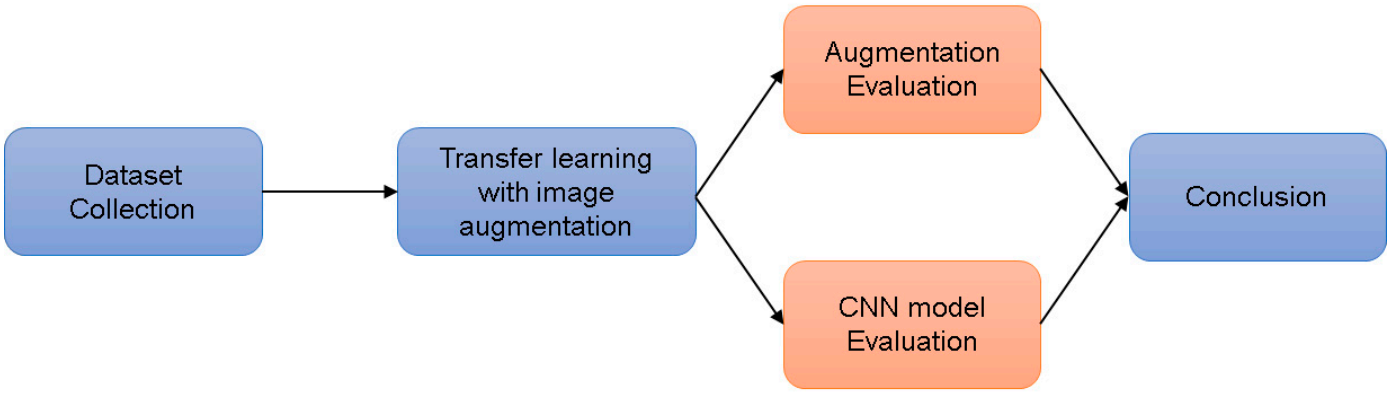

Figure 1. Overview of the research methodology employed in this study.

3. Material

3.1. Dataset

This research utilized five publicly available datasets for HGR: HG14, DLSI, MU HandImages ASL, Sebastian Marcel Static Hand Gestures, and ArASL2018. These datasets were selected to comprehensively evaluate the most suitable geometric transformation for augmenting the HGR dataset. Each dataset possessed unique characteristics such as gesture categories, image background, image size, and color channels. ArabicSL was the most extensive dataset, containing 54,049 images divided into 32 classes, while MU HandImages ASL was the smallest, with only 2425 images divided into 26 classes.

3.1.1. Hand Gesture 14 (HG14) Dataset

Guler et al. [33] created the Hand Gestures 14 (HG14) dataset, containing 14 hand gestures suitable for hand interaction and application control in augmented reality. The dataset includes 14,000 photos with RGB channels and a size of 256 × 256 pixels. Each image has a simple and uniformly colored background, as shown in Figure 2 [33].

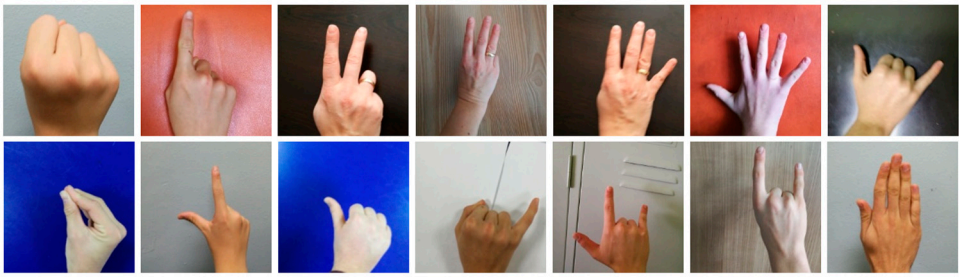

Figure 2. Sample images from the HG14 dataset showcasing the 14 distinct hand gestures.

3.1.2. DLSI (Department de Llenguatges Sistemes Informàtics) Dataset

Alashhab et al. [34] created the DLSI (Department de Llenguatges Sistemes Informàtics) dataset to recognize gestures for visually impaired people. Various smartphone cameras captured indoor and outdoor scenes under realistic conditions. The dataset comprises

12,064 frames divided into 6 gestures, each normalized to 224 × 224 pixels. Figure 3 [34] shows the sample images.

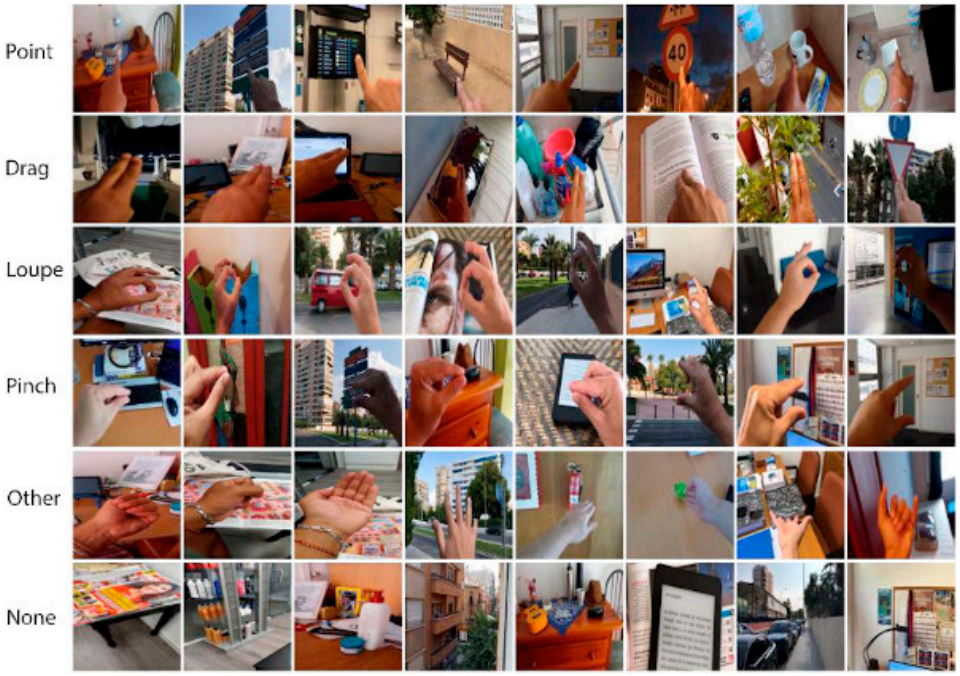

Figure 3. Sample images of the six different hand gestures are included in the DLSI dataset.

3.1.3. Massey University HandImages ASL Dataset

The dataset MU HandImages ASL was created by Barczak et al. at Massey University (MU), New Zealand. It contains 2425 images from 5 individuals, with each hand pose captured in a room with varying lighting conditions and a green screen background. The dataset consists of 26 classes representing standard American Sign Language (ASL) gestures, with black background images and varying pixel sizes depending on the hand pose's shape. Figure 4 [34] shows some sample images from this dataset.

Figure 4. Sample images from the MU HandImages ASL dataset featuring 26 distinct hand gestures commonly used in ASL.

3.1.4. Sebastian Marcel Static Hand Gesture Dataset

The Sebastian Marcel Static Hand Gesture Dataset was used as the training set for developing a neural network model to recognize hand postures in images. Hand gestures

were segmented using space discretization based on face location and body anthropometry. The dataset includes six hand postures (a, b, c, point, five, v) demonstrated by ten individuals captured in uniform and complex backgrounds with varying image sizes, depending on the hand gesture. Figure 5 [34] shows some sample images from this dataset.

Figure 5. Sample images from the Sebastian Marcel Static Hand Gesture Dataset featuring six distinct hand gestures demonstrated by ten individuals.

3.1.5. ArASL2018 Dataset

The ArASL2018 (Arabic Alphabet Sign Language 2018) dataset [35] includes 54,049 grayscale images of 32 hand poses representing the Arabic Alphabet Sign Language. Hand gestures were captured from 40 individuals across different age groups under uniform backgrounds and good lighting conditions. Some image preprocessing was performed to remove noise and center the hand object in the image. Figure 6 [35] shows some sample images from this dataset.

Figure 6. Sample images from the ArASL2018 dataset featuring 32 distinct hand poses representing Arabic Sign Language.

3.2. Image Augmentation

Image augmentation is a technique to avoid overfitting in neural network training by transforming the original data or image using specific techniques [18]. Geometric transformations such as scaling, rotation, translation (shifting), shearing, and flipping can augment image data. These transformations can produce new images from the original ones, helping to generalize the knowledge learned by classifiers, such as neural networks.

Traditional image augmentation involves storing the augmented data on disk alongside the original data, resulting in increased storage requirements, especially for large datasets. An alternative approach called "on-the-fly" augmentation was proposed to address this issue [18]. This approach performs augmentation during training rather than storing augmented images in storage, leading to better knowledge generalization because it produces new data/images in every training epoch. This work used the on-the-fly augmentation strategy to achieve better performance in hand gesture recognition. The flow of on-the-fly augmentation is illustrated in Figure 7.

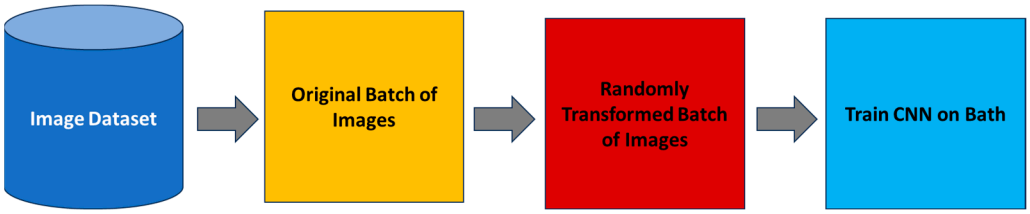

Figure 7. Illustration of the "on-the-fly" image augmentation approach.

As shown in Figure 7, the original image dataset is divided into small batches. Each batch undergoes random geometric transformations before being used for training a machine learning or deep learning algorithm. This technique produces different augmented images in each batch and epoch, allowing for better knowledge generalization.

3.3. Geometric Transformation

This work evaluated all geometric transformations to determine the most suitable for the HGR task. However, it should be noted that certain transformations, such as inverting, may not be ideal for specific image types, such as digit images, which can confuse the numbers 6 and 9.

3.3.1. Image Scaling

Image scaling resizes an input image to a larger or smaller size using a scale factor. Equations (1) [36] and (2) [36] can be used to perform image scaling, where (x, y) are the coordinates of a pixel in the original image, (x', y') are the coordinates of the pixel in the scaled image, and s_x and s_y are the scale factors for the rows and columns of the image, respectively.

$$x' = x \cdot s_x \tag{1}$$

$$y' = y \cdot s_y \tag{2}$$

Interpolation can be used to smooth the edges of the object in the scaled image and maintain the aspect ratio when $s_x = s_y$. Image scaling improves model performance and robustness to input size variations. When an image is scaled up, it is cropped to its original size, while for scaling down, the original size is maintained with space filled using the nearest pixel neighbor technique. Figure 8 shows a sample of a scaled image.

Figure 8. Samples of scaled images using nearest neighbor interpolation.

3.3.2. Image Rotation

Image rotation is a common data augmentation technique in computer vision tasks and involves rotating an image by a certain angle (typically 0 to 360 degrees) to create additional training data. Equations (3) [37] and (4) [37] perform image rotation using (x,y) as the original pixel coordinates and (x',y') as the corresponding pixel coordinates in the rotated image. The rotation angle in radians is represented by θ, and (cx, cy) are the image center coordinates.

$$x' = (x - cx)\cos\theta - (y - cy)\sin\theta + cx \tag{3}$$

$$y' = (x - cx)\sin\theta + (y - cy)\cos\theta + cy \tag{4}$$

Similar to image scaling, image rotation may result in blank areas that need to be filled using interpolation techniques, such as the nearest pixel technique used in this work. Figure 9 provides a sample of image rotation using the nearest pixel technique to interpolate the blank areas around the rotated image.

Figure 9. Sample of image rotation with nearest pixel interpolation.

3.3.3. Image Translation

Image translation shifts an image along the x-axis and y-axis by a certain number of pixels to create additional training data, improving model robustness to position variations. Equations (5) [38] and (6) [38] are used for x-axis and y-axis translation, respectively, where (x,y) are the coordinates of a pixel in the original image, (x',y') are the coordinates of the corresponding pixel in the translated image, and (dx, dy) are the translation offsets.

$$x' = x + dx \tag{5}$$

$$y' = y + dy \tag{6}$$

The nearest pixel interpolation technique also fills blank areas in the translated images. Figure 10 shows a sample of a translated image.

Figure 10. Sample of the translated image using nearest pixel interpolation.

3.3.4. Image Shearing

Image shearing is a technique that skews an image along the x-axis and y-axis by shifting each row or column of pixels by a certain amount based on its y-coordinate or x-coordinate, respectively. This technique helps handle input images captured from different perspectives or angles. Shearing an image along the x-axis and y-axis can be achieved using Equations (7) [39] and (8) [39], where (x,y) are the coordinates of a pixel in the original image, (x',y') are the coordinates of the corresponding pixel in the sheared image, and shx and shy are the shear factors along the x-axis and y-axis, respectively.

$$x' = x + shx \times y \tag{7}$$

$$y' = y + shy \times x \tag{8}$$

Nearest pixel neighbor interpolation is applied to fill the blank area in the sheared images. Figure 11 provides a sample of the sheared image using nearest pixel interpolation.

Figure 11. Sample of the sheared image with nearest pixel interpolation.

3.3.5. Image Flipping

Image flipping reverses the left and right sides of the image. For HGR, only horizontal flipping is used. The formula for horizontal flipping is provided in Equation (9) [40], where (x,y) are the coordinates of a pixel in the original image, x' is the corresponding pixel index in the flipped image, and W is the width of the image.

$$x' = (W - 1) - x \tag{9}$$

To flip the entire image, each row of pixels is reversed from left to right. No interpolation is needed for flipping an image horizontally. Figure 12 shows a sample of horizontally flipped images.

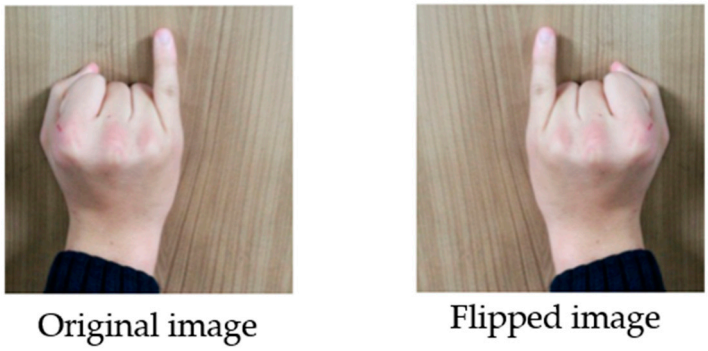

Figure 12. Sample of horizontally flipped image.

3.4. Convolutional Neural Networks (CNNs)

Deep learning (DL) has various architectures, one of which is Convolutional Neural Networks (CNNs), which are known for their effectiveness in image recognition compared to traditional machine learning approaches [41]. The basic idea of the CNN is the technique of image convolution, which combines an input matrix and a kernel matrix to produce a third matrix that represents how one matrix is modified by the other.

A CNN architecture generally consists of two parts: feature extraction and classification [42], as depicted in Figure 13. The feature extraction part applies image convolution to the input image to produce a series of feature maps. These features are then used in the classification part to classify the label of the input image.

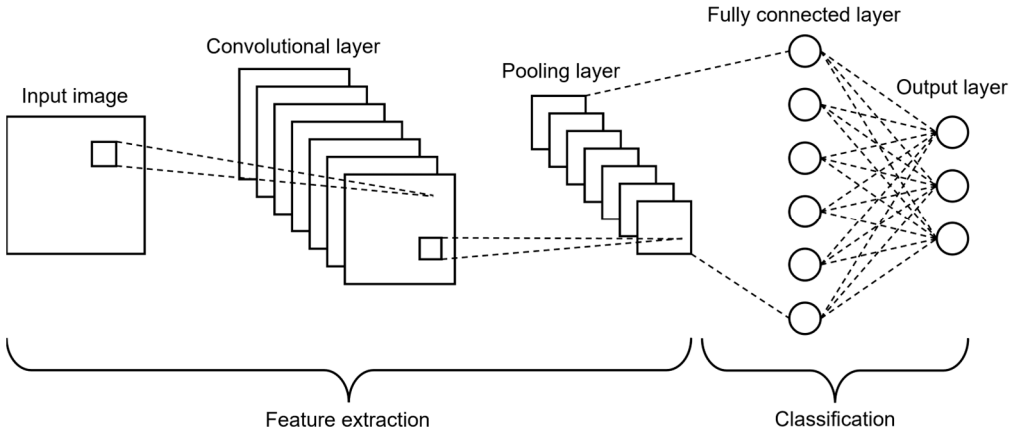

Figure 13. CNN architecture with feature extraction and classification parts.

A CNN applies convolution to produce a set of feature maps. Each filter can detect specific patterns from the input image, such as edges, lines, or corners. The output of the convolutional layer is passed through a non-linear activation function, such as ReLUs (Rectified Linear Units), to introduce non-linearity and learn complex representations of the input image. The ReLU function is defined in Equation (10) [43]:

$$f(x) = max(0,\ x) \tag{10}$$

ReLUs (Rectified Linear Units) are an activation function introduced by [44] and have a strong biological and mathematical underpinning. In 2011, they were demonstrated to further improve the training of deep neural networks. They work by thresholding values at 0, i.e., f(x) = max(0, x). Simply put, they output 0 when $x < 0$, and conversely, they output a linear function when $x \geq 0$.

The pooling layer is used after the convolutional layer to reduce the dimensionality of the feature maps and introduce translational invariance. Max pooling takes the maximum value within a local neighborhood, and average pooling calculates the average value.

Fully connected layers perform a classification task by mapping the previous layer's output to a set of output classes. The network adjusts its weights and biases through a backpropagation algorithm to minimize a loss function that measures the difference between the predicted and true output during training.

3.5. Pre-Trained Neural Networks

To be effective, a CNN model should be sufficiently deep (deep CNN) and trained on large amounts of data to learn various patterns from the dataset [45]. Training a CNN requires a powerful machine with a dedicated Graphics Processing Unit (GPU) for parallel computation and ample memory to store the dataset and CNN parameters during training. Therefore, researchers have developed pre-trained CNN models on large public datasets, such as ImageNet or Microsoft COCO [46]. These pre-trained models can be used to build a CNN-based system for image classification and can be retrained for custom cases using new datasets. This process, called transfer learning, enables pre-trained models to be trained in less time and use less computational power. This work uses three pre-trained CNN models for static hand gesture recognition (HGR) tasks, ResNet50, MobileNetV2, and InceptionV3, due to their excellent performance, as reported in [32].

3.5.1. ResNet50

ResNet50 is a pre-trained CNN model proposed by Microsoft Research in 2015 [41]. It uses Residual Network architecture to solve the problem of vanishing gradients in deep learning. The Residual Network contains residual connections, which add the input of a layer to its output, creating a shortcut connection. Figure 14 shows that the network can skip over layers that might not contribute much to the output.

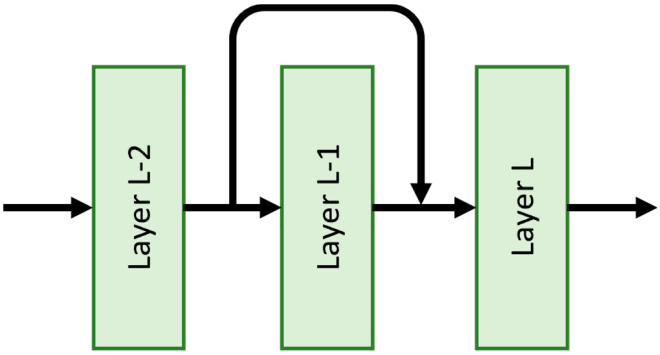

Figure 14. Illustration of a residual connection with the input of layer L-1 added to its output.

The ResNet50 architecture comprises 50 layers, including 49 convolutional layers and 1 fully connected layer, and is divided into 5 convolutional stages. Each stage contains several residual blocks, as shown in Figure 15. The first stage consists of a single convolutional layer that performs a 7×7 convolution with a stride of 2, followed by a max pooling layer with a pool size of 3×3 and a stride of 2. The subsequent stages contain three, four, six, and three residual blocks, each with three convolutional layers using a combination of 1×1, 3×3, and 1×1 convolutions, batch normalization, and ReLU activation. The output of the fifth convolutional stage is processed by a global average pooling layer that averages the feature maps across the spatial dimensions. The resulting output is passed through a fully connected layer with 1000 output units corresponding to the number of classes in the ImageNet dataset.

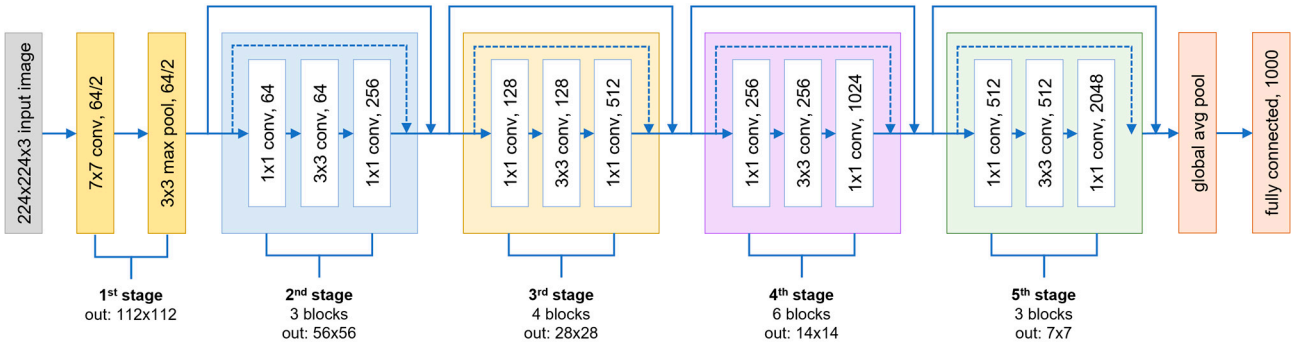

Figure 15. ResNet50 architecture with 49 convolutional layers and 1 fully connected layer.

3.5.2. MobileNetV2

MobileNetV2 is a CNN model proposed by Google in 2018 as an efficient alternative to deep neural networks for deployment on mobile and embedded devices [47]. It uses depthwise separable convolutions, which split the convolution operation into a depthwise convolution and a pointwise convolution, significantly reducing computational complexity and memory usage. MobileNetV2 also introduces several new features that improve its performance while maintaining efficiency. Figure 16 illustrates the working of depthwise separable convolutions.

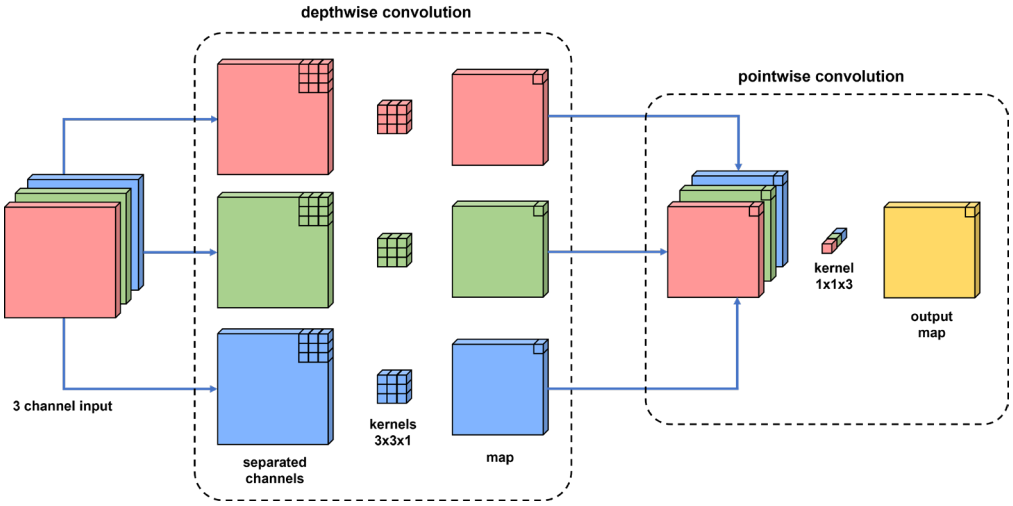

Figure 16. Illustration of depthwise separable convolutions showing the depthwise convolution and pointwise convolution.

The architecture of MobileNetV2 can be divided into three parts: the stem, the body, and the head. The stem has a single convolutional layer that performs a 3×3 convolution with a stride of 2, followed by batch normalization and a non-linear activation function. The body contains a series of inverted residual blocks, each with a depthwise separable convolution, a linear bottleneck, and batch normalization with a non-linear activation function. The MobileNetV2 architecture includes a linear bottleneck that enhances the network's representational power while keeping its computational cost low. The architecture's head consists of a global average pooling layer, a 1×1 convolutional layer, and a fully connected layer that uses a SoftMax activation function. Figure 17 provides a visual representation of the MobileNetV2 architecture.

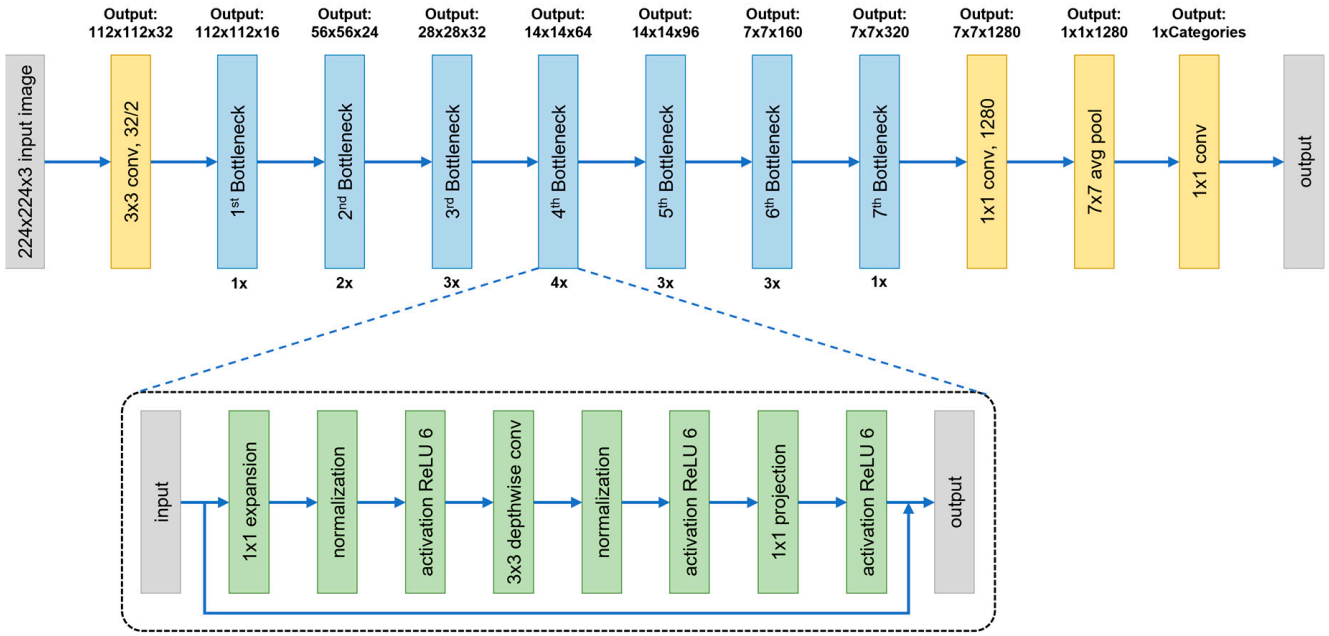

Figure 17. MobileNetV2 architecture with 17 bottleneck layers.

3.5.3. InceptionV3

InceptionV3 is a pre-trained CNN architecture introduced by Google in 2015 [48,49]. It uses multiple filters of different sizes in parallel at each stage of the network to capture features at multiple resolutions and scales, increasing the ability to recognize objects of different sizes and shapes. InceptionV3 also uses factorization to reduce the computational cost of the convolutional layers for more efficient performance. The architecture can be divided into the stem, inception modules, and classification layers [48]. The stem processes the input image and extracts low-level features, while the inception modules perform most of the computation. Each inception module consists of several parallel convolutional layers of different sizes combined using concatenation. Each module's output passes through a factorization layer, reducing feature map channels. A global average pooling layer, followed by a fully connected layer with SoftMax activation, produces the network's final output. The architecture of InceptionV3 is illustrated in Figure 18.

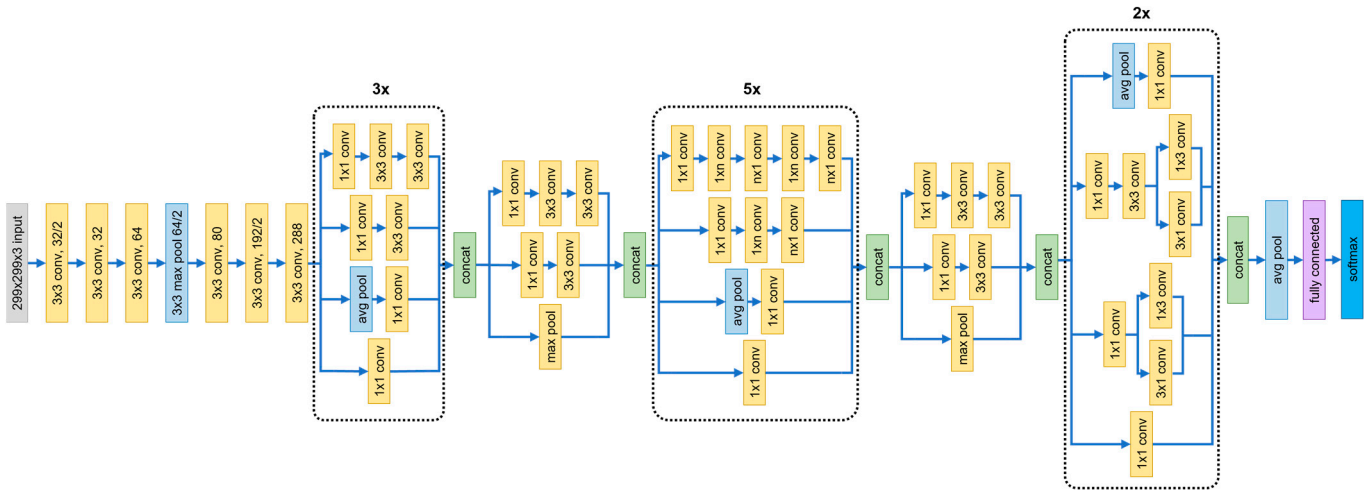

Figure 18. InceptionV3 CNN architecture.

3.6. Programming Tools

In this research using the Python programming language, especially TensorFlow2.6.2, the ImageDataGenerator command serves as a key tool to enhance the performance of models in hand gesture recognition (HGR) tasks. This experiment utilizes each dataset, establishing a solid foundation. By applying transfer learning to pre-trained models, like ResNet50, MobileNetV2, and InceptionV3, the models can comprehend complex features of hand gestures.

The official TensorFlow documentation [50] explains that the ImageDataGenerator plays a central role in image augmentation, exploring geometric transformations like scaling, rotation, translation, shearing, and flipping. The use of commands such as "fit" to calculate internal statistics and "flow" to dynamically generate batches of images creates an efficient and adaptive experimental environment [50]. In TensorFlow, the ImageDataGenerator provides various parameters to control augmentation, such as scaling, rotation, and flipping. The official TensorFlow documentation offers examples of usage and recommended methods. For example:

```
"'python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Creating an instance of ImageDataGenerator with augmentation parameters
datagen = ImageDataGenerator(
rotation_range = 20,
width_shift_range = 0.2,
height_shift_range = 0.2,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True,
fill_mode = 'nearest'
)

# Calculating internal statistics (fit) - assuming x_train is the image data
datagen.fit(x_train)

# Generating batches of images dynamically using flow
generated_images = datagen.flow(x_train, batch_size = 32)
"'
```

Experiments are conducted by trying each pre-trained model on different datasets, with the application of individual and combined augmentations. This structured approach provides a deep understanding of the impact of augmentation on model performance across various datasets. The official TensorFlow documentation, including code examples and parameters, serves as the primary reference source to detail the usage of the ImageDataGenerator and strengthen the experimental framework.

4. Results

4.1. Experimental Setup

This study aims to determine the optimal geometric transformation for image augmentation on the HGR dataset to improve classifier accuracy. To achieve this, it employed an "on-the-fly" augmentation strategy and defined several parameters for random geometric transformations, as shown in Table 1. Each transformation function was applied individually during the training phase of the deep learning algorithm, and the process was repeated three times to ensure consistent performance.

Table 1. Parameter settings for geometric transformations used in image augmentation.

Geometric Transformation	Direction	Parameter Setting
Scaling	Horizontal and vertical	[−20%, 20%]
Rotation	CW and CCW	[−30°, 30°]
Translation	Horizontal and vertical	[−20%, 20%]
Shearing	Horizontal	[−20°, 20°]
Flipping	Horizontal	[True, False]

Transfer learning uses ImageNet pre-trained weights as the initial network's weights in the experiment to avoid complex and high computation during learning. The optimization of weights only occurs in the classification layer of the pre-trained model to optimize the networks in the fully connected layers. Adaptive Moment Estimation (ADAM), an optimization algorithm, is employed to enhance the training process and avoid gradient vanishing during training [51]. The pre-trained network is retrained for 50 epochs with a batch size of 32. To maintain the ImageNet pre-trained weights, network training is performed by freezing all the layers in the feature extraction part. The performance of the networks is evaluated using the "accuracy" metric.

The experiment uses Python programming language with several libraries, such as TensorFlow, Matplotlib, and NumPy, on a personal computer with the specifications listed in Table 2.

Table 2. Hardware and software specifications for the experiment.

Hardware/Software	Specification
Processor (CPU)	Intel Core i5-9300H @2.40 GHz
Memory (RAM)	32 GB DDR4
Graphics Processing Unit (GPU)	Nvidia GTX 1660 Ti—6GB vRAM
Operating system	Windows 11
Python version	3.6.13
Cuda/CuDNN version	11.0/8.0

4.2. Dataset Preparation

In this study, we used five publicly available datasets of HGR, as described in Section 2. These datasets were used to evaluate the effectiveness of various experiment scenarios, as summarized in Table 3. To train our networks, we randomly split each dataset into three parts for training, evaluation, and testing, respectively, with a distribution of 60:20:20. To augment the training data, we used the ImageDataGenerator module TensorFlow [50]. It is important to note that only the training data were augmented.

Table 3. Specification of datasets used for the experiment.

Dataset	Number of Data	Number of Classes	Images Size	Image Background
DLSI	12,064	6	224 × 224	complex
HG14	14,000	14	256 × 256	uniform
MU HandImages ASL	2425	26	vary	uniform
Sebastian Marcel	5531	6	vary	uniform and complex
ArASL2018	54,049	32	64 × 64	uniform

In this work, it was observed that specific parameters in the HGR dataset could impact the classification performance, such as input size and image background. As shown in Table 3, each dataset had different sizes, so all input images were resized to a uniform size of 224 × 224 pixels with three color channels (RGB) commonly used in pre-trained CNN models like VGG, ResNet, and Inception.

The image background was classified as uniform or complex based on the dominance of a single color or simple texture versus multiple colors, textures, or objects. While a

uniform background is easier to recognize and more accurate, it may not represent real-life scenarios. Conversely, a complex background may be more challenging to recognize but is more representative of real-life situations. Using HGR datasets with uniform and complex backgrounds, we could analyze the impact of geometric-based augmentation on background variations.

4.3. Experimental Results

The experiments were conducted based on the scenarios explained in the previous section, which are divided into two parts. The first part involves a single augmentation experiment to observe which geometric augmentation (scaling, rotation, translation, shearing, and flipping) can lead to the best performance. The second part involves a combined augmentation experiment to observe whether image augmentation using all five geometric transformations produces better performance than a single one. As the datasets have a different number of classes and are balanced; the only performance metric used is accuracy.

4.3.1. Results on Single Augmentation

The first experiment involved using a pre-trained ResNet50 model that was retrained on the five HGR datasets. Each input image was augmented using the ImageDataGenerator module from the TensorFlow library before being fed into the pre-trained model for learning. Overall, ResNet50 performed very well for all datasets and geometric transformations, achieving accuracies of over 95% for almost all experiment scenarios. Specifically, ResNet50 achieved the highest accuracy when using the DLSI and HG14 datasets, with average accuracies of 97.67% and 97.47%, respectively, for all geometric transformations. This result is surprising because the DLSI dataset has a complex image background. On the other hand, the lowest accuracy was obtained when using the Sebastian Marcel dataset, with an average accuracy of 94.98%. The performance of ResNet50 on all datasets is summarized in Table 4.

Table 4. Classification accuracy results of geometric image augmentation using ResNet50 architecture.

Dataset	Geometric Transformations—Accuracy (%)					Dataset Average Accuracy (%)
	Scaling	Rotation	Translation	Shearing	Flipping	
DLSI	97.86	97.37	97.47	97.86	97.77	**97.67**
HG14	97.07	97.46	98.32	97.18	97.32	**97.47**
MU HandImages ASL	95.14	97.26	96.05	97.26	97.57	**96.66**
ArASL2018	96.61	94.66	96.04	97.26	97.15	**96.34**
Sebastian Marcel	93.60	95.07	96.06	94.58	95.57	**94.98**
Geometric Avg. Accuracy (%)	**96.06**	**96.36**	**96.79**	**96.83**	**97.08**	

Based on the results presented in Table 4, horizontal flipping achieved the highest accuracy among the five geometric transformations, with an average accuracy of 97.08%. Furthermore, image shearing and translation produced better results than scaling and rotation.

Moving on to the second experiment, as shown in Table 5, MobileNetV2 performed worse than ResNet50 for all datasets. The MU HandImage ASL dataset achieved the highest accuracy among all datasets, with significant differences from the other datasets of up to 38%. However, MobileNetV2 performed poorly in classifying the DLSI dataset with a complex image background. Unlike ResNet50, shearing transformation produced the best results for all datasets, with an overall accuracy of 78.85%, followed by horizontal flipping, with an average accuracy of 74.72%.

Table 5. Results of geometric image augmentation on the MobileNetV2 architecture.

Dataset	Geometric Transformations—Accuracy (%)					Dataset Average Accuracy (%)
	Scaling	Rotation	Translation	Shearing	Flipping	
DLSI	73.63	73.78	69.96	79.79	76.66	**74.76**
HG14	59.18	57.75	59.00	65.18	60.39	**60.30**
MU HandImages ASL	92.40	90.88	91.49	95.74	95.74	**93.25**
ArASL2018	69.21	59.53	63.83	84.09	74.79	**70.29**
Sebastian Marcel	65.52	61.08	63.05	69.46	66.01	**65.02**
Geometric Avg. Accuracy (%)	**71.99**	**68.60**	**69.47**	**78.85**	**74.72**	

Table 6 shows the results of using the InceptionV3 architecture for geometric image augmentation. Similar to MobileNetV2, the MU HandImage ASL dataset achieved the highest accuracy. The shearing operation resulted in the highest accuracy compared to other geometric transformations for all datasets. However, InceptionV3 struggled to recognize the hand gestures in the HG14 dataset, which had the lowest accuracy. This is surprising because HG14 uses a uniform background in each image, and the difference in accuracy compared to other datasets is significant.

Table 6. Results of geometric image augmentation on the InceptionV3 architecture.

Dataset	Geometric Transformations—Accuracy (%)					Dataset Average Accuracy (%)
	Scaling	Rotation	Translation	Shearing	Flipping	
DLSI	67.28	65.44	60.92	71.70	69.27	**66.92**
HG14	48.57	41.54	39.43	49.11	40.25	**43.78**
MU HandImages ASL	89.06	86.63	82.67	94.22	89.06	**84.34**
ArASL2018	82.93	76.75	67.33	85.75	74.37	**70.47**
Sebastian Marcel	70.94	73.89	67.49	74.38	72.41	**74.07**
Geometric Avg. Accuracy (%)	**71.76**	**68.85**	**63.57**	**75.03**	**69.07**	

4.3.2. Results of Combined Augmentation

This study performed experiments to compare the performance of pre-trained models trained with single and combined geometric transformations. The same experimental setup as before was used with three repetitions for each dataset to ensure consistency of model performance. Table 7 shows that single augmentation yielded better results than combined augmentation for every pre-trained model. The most significant decreases in accuracy were observed in MobileNetV2 and InceptionV3, which experienced drops of 11.54% and 18.00%, respectively. The worst accuracy was obtained for HG14 and ArASL2018 datasets when using MobileNetV2 and InceptionV3 with combined augmentation, with an accuracy below 50%. However, MU HandImages ASL maintained good accuracy for each pre-trained model using single or combined augmentation.

Table 7. Comparison of accuracies between single and combined augmentation methods.

Dataset	ResNet50 (%)		MobileNetV2 (%)		InceptionV3 (%)	
	Single	Combined	Single	Combined	Single	Combined
DLSI	97.67	97.96	74.76	65.29	66.92	52.09
HG14	97.47	96.61	60.30	48.86	43.78	28.82
MU HandImages ASL	96.66	93.62	93.25	94.22	84.34	76.60
ArASL2018	96.34	92.41	70.29	39.38	70.47	18.47
Sebastian Marcel	94.98	95.57	65.02	58.13	74.07	74.38
Avg. Accuracy (%)	**96.62**	**95.23**	**72.72**	**61.18**	**67.92**	**50.07**

5. Discussion

The experimental discussion in this research presents noteworthy findings that can serve as a foundation for further research and development in hand gesture recognition (HGR). Analyzing the documented experimental results in Sections 4–6 reveals that image shearing holds the most significant influence on the classification accuracy among the five geometric transformations used for image augmentation. It is noteworthy that despite the accuracy value of image shearing in ResNet50 being lower compared to the flipping transformation, MobileNet, and Inception, in contrast, it achieved the highest accuracy values.

This observed significance can be attributed to the technical aspects of image shearing, wherein its ability to handle variations in object perspective and other specific characteristics renders it a crucial augmentation technique for enhancing the accuracy of hand gesture recognition models.

The implication of these findings is that constructing a classification model capable of effectively managing variations in object perspective is crucial for achieving high accuracy in HGR. Furthermore, image flipping emerges as the second most influential geometric transformation for the HGR task, as evidenced by the relatively higher accuracy observed in both image shearing and flipping, as indicated in Figure 21. This underscores the importance of developing an HGR model that can adeptly handle both variations in object perspective and the reflection of gestures by the right or left hand. It is noteworthy that while image scaling can impact classification accuracy, its effect is comparatively lower than image shearing and flipping.

The use of image rotation as an augmentation method in hand gesture recognition (HGR) datasets is generally applicable, but its effectiveness depends on the types of gestures present in the dataset. Some gestures may require hand position rotation to differentiate between distinct signs. For instance, in the MU HandImage ASL (Massey dataset), the gesture for the letter "i" is nearly identical to the letter "j", and the gesture for the letter "z" is identical to the number "1", as shown in Figure 19. In such cases, it is advisable to carefully consider the use of image rotation or even exclude it from augmentation operations to avoid introducing ambiguity into the dataset.

Figure 19. The identical gestures between the number "1" and the letter "z" shown in (**a**) and the letters "i" and "j" shown in (**b**) in the MU HandImage ASL dataset are only distinguished by hand position. Therefore, rotation can be excluded from the augmentation operations.

Similar to rotation, Figure 20 illustrates that image translation, particularly in the HG14 dataset, can produce ambiguously augmented data, such as in the HG14 dataset, where the horizontal translation of gesture number "6" may be identical to gesture number "9". Similarly, gesture number "11" may be highly like gesture number "12" when shifted to the left. To handle this issue, translation can be excluded from augmentation, or the range of translation can be limited.

Figure 20. Similar hand gestures in the HG14 dataset due to translational transformation: (**a**) "6" and "9" and (**b**) "11" and "12".

Figure 21 provides additional insights, confirming that image shearing and flipping are effective techniques for augmenting static hand gesture recognition (HGR) datasets. This effectiveness holds true across diverse datasets and pre-trained models. While image scaling is a viable option, Figure 22 highlights a cautionary note—using an excessive number of transformations can lead to overly complex images, potentially causing misclassification. Consequently, a more practical approach for image augmentation in HGR tasks using Convolutional Neural Networks (CNNs) may involve combining two geometric transformations.

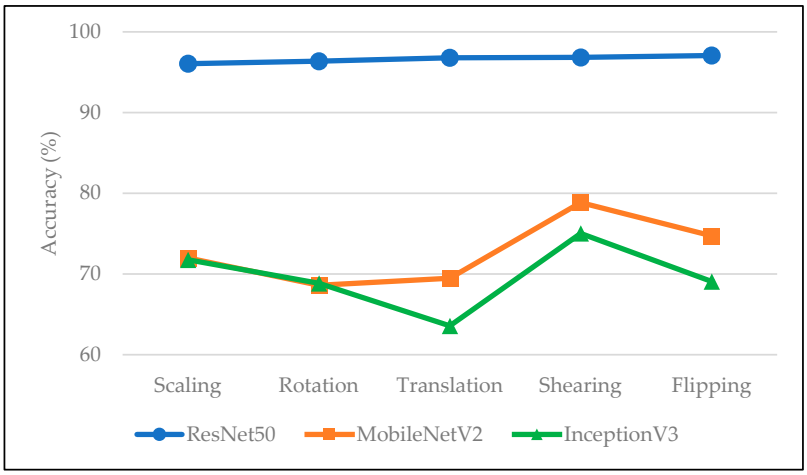

Figure 21. Comparison of classification accuracy among different geometric transformations using different pre-trained models.

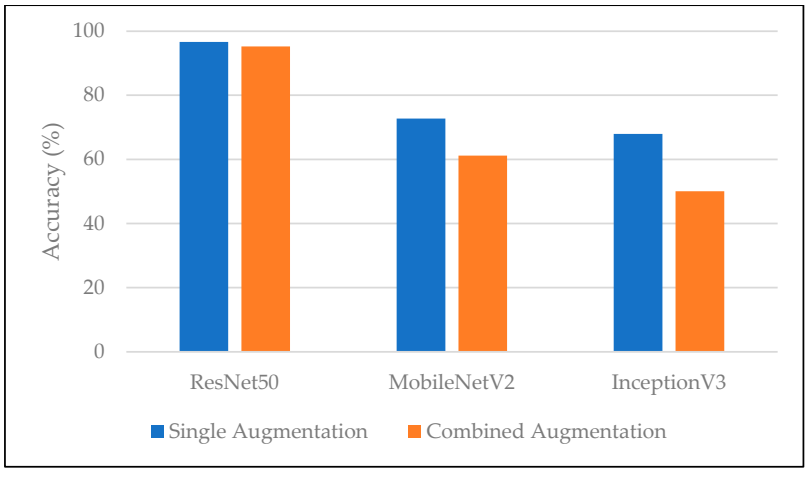

Figure 22. Comparison of classification performance between single and combined geometric augmentations on different pre-trained models.

This experiment reveals that ResNet50 outperforms MobileNetV2 and InceptionV3 in static HGR datasets. Figure 22 demonstrates the superior performance of ResNet50, a popular CNN architecture, compared to MobileNetV2 and InceptionV3. The notable advantages of ResNet50, including its less complex network that demands less computing power and memory, have significant practical implications for real-world applications. In the development of hand gesture recognition applications for devices with limited computational resources, choosing ResNet50 can result in faster and more energy-efficient models. This is particularly crucial in scenarios such as mobile devices, where optimizing resource usage is paramount for a seamless user experience.

6. Conclusions

Based on the results of the experiments conducted for static hand gesture recognition using Convolutional Neural Network (CNN) models, several conclusions can be drawn to guide the development of more efficient methods for this task.

Firstly, in the context of geometric transformations, it was found that the use of shearing and flipping had the most significant impact on improving the model's accuracy. These transformations help the model better cope with variations in object perspectives, indicating that building a classification model capable of handling changes in viewpoint is a crucial aspect of static hand gesture recognition.

Secondly, in choosing a CNN model, ResNet50 consistently proved to outperform MobileNetV2 and InceptionV3. Although ResNet50 may require more computational power, the advantage of high-accuracy results suggests its importance in selecting a model for this task.

However, the conclusions also highlight that overly complex geometric transformations can harm the model's performance, especially for MobileNetV2 and InceptionV3. Excessive geometric transformations may involve using too many types of transformations, making the model struggle to understand actual patterns, as each training example experiences significant variation. Therefore, a balance is needed between performance improvement and real-world representation by selecting transformations suitable for the dataset's characteristics.

Moreover, future research may consider color modifications in addition to geometric transformations. Further understanding of how to optimize image augmentation methods, including color variations, can significantly contribute to improving model performance, especially in situations where color variations may affect image interpretation.

As a direction for future research, this study can be expanded by combining multiplex geometric transformations with color modifications, creating a more holistic and effective image augmentation framework. Additionally, further exploration can be conducted in situations where there is a lack of data or significant variation between datasets. Integration with transfer learning techniques and the development of more complex models can also be an interesting focus of research. Thus, this study provides a foundation for further exploration in enhancing the efficiency and effectiveness of static hand gesture recognition using a CNN-based approach.

Author Contributions: Conceptualization, B.-A.A.; methodology, B.-A.A.; software, B.-A.A.; validation, B.-A.A. and C.-T.C.; formal analysis, B.-A.A.; investigation, B.-A.A.; resources, B.-A.A.; data curation, B.-A.A.; writing—original draft preparation, B.-A.A.; writing—review and editing, B.-A.A. and C.-T.C.; visualization, B.-A.A.; supervision, C.-T.C. and J.-S.C.; project administration, J.-S.C.; funding acquisition, J.-S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science and Technology Council, Taiwan grant number NSTC 112-2221-E-218-017.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets in this work can be accessed through the link provided in Section 2.

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

References

1. Lee, C.; Kim, J.; Cho, S.; Kim, J.; Yoo, J.; Kwon, S. Development of Real-Time Hand Gesture Recognition for Tabletop Holographic Display Interaction Using Azure Kinect. *Sensors* **2020**, *20*, 4566. [CrossRef]
2. Ekneling, S.; Sonestedt, T.; Georgiadis, A.; Yousefi, S.; Chana, J. Magestro: Gamification of the Data Collection Process for Development of the Hand Gesture Recognition Technology. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Munich, Germany, 16–20 October 2018; pp. 417–418.
3. Bai, Z.; Wang, L.; Zhou, S.; Cao, Y.; Liu, Y.; Zhang, J. Fast Recognition Method of Football Robot's Graphics From the VR Perspective. *IEEE Access* **2020**, *8*, 161472–161479. [CrossRef]
4. Nooruddin, N.; Dembani, R.; Maitlo, N. HGR: Hand-Gesture-Recognition Based Text Input Method for AR/VR Wearable Devices. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 744–751.
5. Zhao, J.; An, R.; Xu, R.; Lin, B. Comparing Hand Gestures and a Gamepad Interface for Locomotion in Virtual Environments. *Int. J. Hum.-Comput. Stud.* **2022**, *166*, 102868. [CrossRef]
6. Mezari, A.; Maglogiannis, I. An Easily Customized Gesture Recognizer for Assisted Living Using Commodity Mobile Devices. *J. Healthc. Eng.* **2018**, *2018*, 3180652. [CrossRef] [PubMed]
7. Roberge, A.; Bouchard, B.; Maître, J.; Gaboury, S. Hand Gestures Identification for Fine-Grained Human Activity Recognition in Smart Homes. In *Procedia Computer Science*; Elsevier B.V.: Amsterdam, The Netherlands; Liara Laboratory, University of Quebec, Chicoutimi 555 Boul. Universite: Saguenay, QC, Canada, 2022; Volume 201, pp. 32–39.
8. Huang, X.; Hu, S.; Guo, Q. Multi-Object Recognition Based on Improved YOLOv4. In Proceedings of the 2021 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS), Chengdu, China, 17–18 December 2021; pp. 1–4.
9. Kaczmarek, W.; Panasiuk, J.; Borys, S.; Banach, P. Industrial Robot Control by Means of Gestures and Voice Commands in Off-Line and On-Line Mode. *Sensors* **2020**, *20*, 6358. [CrossRef]
10. Neto, P.; Simão, M.; Mendes, N.; Safeea, M. Gesture-Based Human-Robot Interaction for Human Assistance in Manufacturing. *Int. J. Adv. Manuf. Technol.* **2019**, *101*, 119–135. [CrossRef]
11. Ding, I.-J.; Su, J.-L. Designs of Human–Robot Interaction Using Depth Sensor-Based Hand Gesture Communication for Smart Material-Handling Robot Operations. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2023**, *237*, 392–413. [CrossRef]
12. Young, G.; Milne, H.; Griffiths, D.; Padfield, E.; Blenkinsopp, R.; Georgiou, O. Designing Mid-Air Haptic Gesture Controlled User Interfaces for Cars. *Proc. ACM Hum.-Comput. Interact.* **2020**, *4*, 1–23. [CrossRef]
13. Qian, X.; Ju, W.; Sirkin, D.M. Aladdin's Magic Carpet: Navigation by in-Air Static Hand Gesture in Autonomous Vehicles. *Int. J. Hum.–Comput. Interact.* **2020**, *36*, 1912–1927. [CrossRef]
14. Devineau, G.; Moutarde, F.; Xi, W.; Yang, J. Deep Learning for Hand Gesture Recognition on Skeletal Data. In Proceedings of the 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China, 15–19 May 2018; pp. 106–113.
15. Wang, J.; Liu, T.; Wang, X. Human Hand Gesture Recognition with Convolutional Neural Networks for K-12 Double-Teachers Instruction Mode Classroom. *Infrared Phys. Technol.* **2020**, *111*, 103464. [CrossRef]
16. Khoh, W.H.; Pang, Y.H.; Teoh, A.B.J.; Ooi, S.Y. In-Air Hand Gesture Signature Using Transfer Learning and Its Forgery Attack. *Appl. Soft Comput.* **2021**, *113*, 108033. [CrossRef]
17. Khosla, C.; Saini, B.S. Enhancing Performance of Deep Learning Models with Different Data Augmentation Techniques: A Survey. In Proceedings of the 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 17–19 June 2020; pp. 79–85.
18. Shorten, C.; Khoshgoftaar, T.M. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
19. Kalaivani, S.; Asha, N.; Gayathri, A. Geometric Transformations-Based Medical Image Augmentation. In *GANs for Data Augmentation in Healthcare*; Solanki, A., Naved, M., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 133–141, ISBN 978-3-031-43204-0.
20. Islam, M.Z.; Hossain, M.S.; ul Islam, R.; Andersson, K. Static Hand Gesture Recognition Using Convolutional Neural Network with Data Augmentation. In Proceedings of the 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Spokane, WA, USA, 30 May–2 June 2019; pp. 324–329.
21. Bousbai, K.; Merah, M. Hand Gesture Recognition Using Capabilities of Capsule Network and Data Augmentation. In Proceedings of the 2022 7th International Conference on Image and Signal Processing and their Applications (ISPA), Mostaganem, Algeria, 8–9 May 2022; pp. 1–5.

22. Alani, A.A.; Cosma, G.; Taherkhani, A.; McGinnity, T.M. Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation. In Proceedings of the 2018 4th International Conference on Information Management (ICIM), Oxford, UK, 25–27 May 2018; pp. 5–12.
23. Zhou, W.; Chen, K. A Lightweight Hand Gesture Recognition in Complex Backgrounds. *Displays* **2022**, *74*, 102226. [CrossRef]
24. Galdran, A.; Alvarez-Gila, A.; Meyer, M.I.; Saratxaga, C.L.; Araújo, T.; Garrote, E.; Aresta, G.; Costa, P.; Mendonça, A.M.; Campilho, A. Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis. *arXiv* **2017**, arXiv:1703.03702.
25. Tan, Y.S.; Lim, K.M.; Lee, C.P. Hand Gesture Recognition via Enhanced Densely Connected Convolutional Neural Network. *Expert Syst. Appl.* **2021**, *175*, 114797. [CrossRef]
26. Taylor, L.; Nitschke, G. Improving Deep Learning with Generic Data Augmentation. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; pp. 1542–1547.
27. Motamed, S.; Rogalla, P.; Khalvati, F. Data Augmentation Using Generative Adversarial Networks (GANs) for GAN-Based Detection of Pneumonia and COVID-19 in Chest X-Ray Images. *Inform. Med. Unlocked* **2021**, *27*, 100779. [CrossRef] [PubMed]
28. Rajeev, C.; Natarajan, K. Data Augmentation in Classifying Chest Radiograph Images (CXR) Using DCGAN-CNN. In *GANs for Data Augmentation in Healthcare*; Solanki, A., Naved, M., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 91–110. ISBN 978-3-031-43204-0.
29. Farahanipad, F.; Rezaei, M.; Nasr, M.S.; Kamangar, F.; Athitsos, V. A Survey on GAN-Based Data Augmentation for Hand Pose Estimation Problem. *Technologies* **2022**, *10*, 43. [CrossRef]
30. Saxena, D.; Cao, J. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *ACM Comput. Surv. CSUR* **2021**, *54*, 1–42. [CrossRef]
31. Ciano, G.; Andreini, P.; Mazzierli, T.; Bianchini, M.; Scarselli, F. A Multi-Stage GAN for Multi-Organ Chest X-Ray Image Generation and Segmentation. *Mathematics* **2021**, *9*, 2896. [CrossRef]
32. Avianto, D.; Harjoko, A.; Afiahayati. CNN-Based Classification for Highly Similar Vehicle Model Using Multi-Task Learning. *J. Imaging* **2022**, *8*, 293. [CrossRef]
33. Güler, O.; Yücedağ, İ. Hand Gesture Recognition from 2D Images by Using Convolutional Capsule Neural Networks. *Arab. J. Sci. Eng.* **2022**, *47*, 1211–1225. [CrossRef]
34. Alashhab, S.; Gallego, A.J.; Lozano, M.Á. Efficient Gesture Recognition for the Assistance of Visually Impaired People Using Multi-Head Neural Networks. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105188. [CrossRef]
35. Latif, G.; Mohammad, N.; Alghazo, J.; AlKhalaf, R.; AlKhalaf, R. ArASL: Arabic Alphabets Sign Language Dataset. *Data Brief* **2019**, *23*, 103777. [CrossRef] [PubMed]
36. Lecture—Image Processing: Geometric Operations—Scaling | WueCampus. Available online: https://wuecampus.uni-wuerzburg.de/moodle/mod/book/view.php?id=958001&chapterid=10072 (accessed on 17 November 2023).
37. Lecture—Image Processing: Geometric Operations—Rotation | WueCampus. Available online: https://wuecampus.uni-wuerzburg.de/moodle/mod/book/view.php?id=958001&chapterid=10071 (accessed on 17 November 2023).
38. Lecture—Image Processing: Geometric Operations—Translation | WueCampus. Available online: https://wuecampus.uni-wuerzburg.de/moodle/mod/book/view.php?id=958001&chapterid=10067 (accessed on 17 November 2023).
39. Shearing in 2D Graphics. GeeksforGeeks 2020. Available online: https://www.geeksforgeeks.org/shearing-in-2d-graphics/ (accessed on 17 November 2023).
40. Lecture—Image Processing: Geometric Operations—Mirroring | WueCampus. Available online: https://wuecampus.uni-wuerzburg.de/moodle/mod/book/view.php?id=958001&chapterid=10073 (accessed on 17 November 2023).
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
42. Phung, V.H.; Rhee, E.J. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Appl. Sci.* **2019**, *9*, 4500. [CrossRef]
43. Agarap, A.F. Deep Learning Using Rectified Linear Units (ReLU). *arXiv* **2019**, arXiv:1803.08375.
44. Hahnloser, R.H.R.; Sarpeshkar, R.; Mahowald, M.A.; Douglas, R.J.; Seung, H.S. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. *Nature* **2000**, *405*, 947–951. [CrossRef]
45. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]
46. Subburaj, S.; Murugavalli, S. Survey on Sign Language Recognition in Context of Vision-Based and Deep Learning. *Meas. Sens.* **2022**, *23*, 100385. [CrossRef]
47. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
48. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.

49. Szegedy, C.; 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, 7–12 June 2015; pp. 1–9.
50. Tf.Keras.Preprocessing.Image.ImageDataGenerator | TensorFlow v2.14.0. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (accessed on 13 November 2023).
51. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.