applied sciences

MDPI

Article

Human Posture Detection Using Image Augmentation and Hyperparameter-Optimized Transfer Learning Algorithms

Roseline Oluwaseun Ogundokun, Rytis Maskeliūnas and Robertas Damaševičius *

Faculty of Informatics, Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania

* Correspondence: robertas.damasevicius@ktu.lt

Abstract: With the advancement in pose estimation techniques, human posture detection recently received considerable attention in many applications, including ergonomics and healthcare. When using neural network models, overfitting and poor performance are prevalent issues. Recently, convolutional neural networks (CNNs) were successfully used for human posture recognition from human images due to their superior multiscale high-level visual representations over hand-engineering low-level characteristics. However, calculating millions of parameters in a deep CNN requires a significant number of annotated examples, which prohibits many deep CNNs such as AlexNet and VGG16 from being used on issues with minimal training data. We propose a new three-phase model for decision support that integrates CNN transfer learning, image data augmentation, and hyperparameter optimization (HPO) to address this problem. The model is used as part of a new decision support framework for the optimization of hyperparameters for AlexNet, VGG16, CNN, and multilayer perceptron (MLP) models for accomplishing optimal classification results. The AlexNet and VGG16 transfer learning algorithms with HPO are used for human posture detection, while CNN and Multilayer Perceptron (MLP) were used as standard classifiers for contrast. The HPO methods are essential for machine learning and deep learning algorithms because they directly influence the behaviors of training algorithms and have a major impact on the performance of machine learning and deep learning models. We used an image data augmentation technique to increase the number of images to be used for model training to reduce model overfitting and improve classification performance using the AlexNet, VGG16, CNN, and MLP models. The optimal combination of hyperparameters was found for the four models using a random-based search strategy. The MPII human posture datasets were used to test the proposed approach. The proposed models achieved an accuracy of 91.2% using AlexNet, 90.2% using VGG16, 87.5% using CNN, and 89.9% using MLP. The study is the first HPO study executed on the MPII human pose dataset.

Keywords: data augmentation; CNN; dropout; transfer learning; human pose detection; hyperparameter optimization; decision support

check for updates

Citation: Ogundokun, R.O.; Maskeliūnas, R.; Damaševičius, R. Human Posture Detection Using Image Augmentation and Hyperparameter-Optimized Transfer Learning Algorithms. *Appl. Sci.* **2022**, *12*, 10156. https://doi.org/10.3390/app121910156

Academic Editor: Maria Rizzi

Received: 1 September 2022
Accepted: 3 October 2022
Published: 10 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the availability of large-scale category-level training data, such as ImageNet [1], and an effective overfitting avoidance technique ("dropout") [2], deep convolutional neural networks (CNN) defeated all known methods in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 [3]. The fundamental benefit of CNN for image classification is that the entire system is trained from raw pixels to final categories, eliminating the need to construct an appropriate feature extractor manually. The key drawbacks of CNN are: (i) for weight parameter learning, many labeled training samples are essential; and (ii) a strong Graphics Processing Unit (GPU) is required to accelerate the learning process. The popular deep CNN design (shown in Figure 1) is based on [4] and consists of 5 convolutional layers, 3 fully connected layers, and a final soft-max classifier with approximately 60 million parameters. Deeper convolutional networks with more parameters, such as 16 and 19 layers [5], and 22 layers [6], can achieve superior performance. Even if the

over-fitting preventive strategy is used, learning that many parameters from just thousands of training samples will result in substantial over-fitting. As a result, it is difficult to fit a deep CNN to a small dataset while maintaining equivalent performance to a large dataset. Transfer learning, which drops the classifier layer of a pre-trained CNN and fine-tunes it on the target dataset, is a typical approach for employing deep CNNs on short datasets. This is an efficient strategy in practice. While this strategy can be used to solve a specific problem, it leaves us with some unanswered questions: how to choose the fine-tuning strategy, how to set the hyperparameters (e.g., learning rate) for network fine-tuning, and additional approaches to increase the network performance [7].

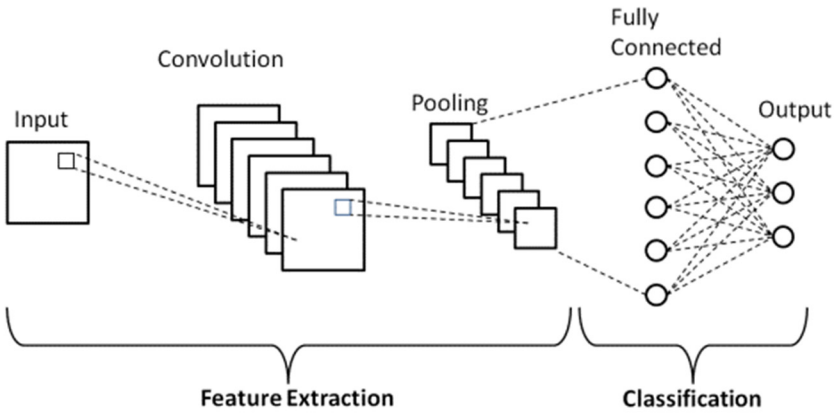


Figure 1. Deep CNN Architecture [8].

Several manuscripts have used machine learning and deep learning methods to solve the human posture classification problem [9,10], but none applied hyperparameter optimization (HPO) with algorithms to find the best hyperparameters that will induce the most excellent classification accuracy for the machine learning or deep learning algorithms used (as far as we know). This set of hyperparameters is not the same for every classification task and changes according to the nature of the medical problem, which is complex by design. When a critical data analysis of medical data is required to identify hidden links or abnormalities that are not evident to humans, machine learning technologies are being employed in healthcare for computational decision-making [11]. Decision support systems combined with various artificial intelligence (AI) methods have been used for supporting various medical decisions while analyzing complex biomedical signals and images such as echocardiograms [12], magnetic resonance images [13] and chest x-rays, and computed tomography images [14].

Usually, huge amounts and complicated medical data, reports, and images must be analyzed more quickly but with higher accuracy. It is challenging to implement algorithms to carry out such jobs in and of themselves, but it is even more difficult to improve algorithm accuracy while reducing execution time. Especially, hyperparameter optimization may require a much larger overhead, because multiple training rounds and evaluations of machine/deep learning algorithms for hyperparameter-optimized decision support are needed. However, most of these methods solely concentrate on feature selection while focusing specifically on the issues of underfitting and overfitting. The model can perform well on both datasets, i.e., training data and testing data, if overfitting and underfitting are prevented. Overfitting of the training data is frequently caused by irrelevant features and improper (i.e., suboptimal) model design [15]. Most models proposed in the literature are not generalizable, which raises the need for optimizing models that adapt well to new, previously unavailable medical data [16]. As hyperparameter optimization is often neglected in the validation of decision support systems, the significance of its effects remains unreported [17]. Moreover, there is a lack of systematic studies on the effect of hyperparameters on the accuracy and robustness of the AI models used for decision support [18].

This paper aims to propose a rigorous methodology for CNN transfer learning HPO for human posture image classification. For this, the random search approach was adopted to create recommended rankings for filter sizes for the convolutional layers and dense layers of the four models and the learning rate values for the optimizer used in the models. Additionally, the dataset used in the study includes the MPII human pose images obtained from the Kaggle repository. Unlike current approaches, the models and techniques utilized in this proposed system are more favorable since they enhance and optimize the classification of human posture using deep learning algorithms.

The remaining part of the article is prearranged as follows: Section 2 presents the related works. Section 3 describes the two HPO models used in the study. This section also presented the materials and methods used in the study. Section 4 presents the experimental results and discussions. The discussion is presented in Section 5, and the article is concluded with future works suggested in Section 6.

2. Literature Review

In this article, we described the utilization of HPO of deep learning algorithms and data augmentation with deep learning models to solve image classification problems. It is essential to shedding light on earlier related works on image classification using deep learning algorithms. Machine learning in limited datasets is aided by data augmentation approaches [19–23]. This is because the creation of false pictures immediately helps to increase the deep learning model's capacity for generalization and therefore lowers the risk of overfitting [19,20]. One of the difficulties with data augmentation in this regard is determining which transformations (such as zoom, rotation, and flip) will be applied to the images [24–28]. This topic may be categorized as a HPO problem in terms of machine learning [29–35]. Some research in the literature has looked at the impact of data augmentation and HPO combinations in a variety of applications, including plant classification [36], transmission line inspection [37], and the COVID-19 diagnosis procedure in chest radiographic imaging [38]. Hyperparameter optimization was also applied in bioinformatics to optimize SVM classification [39], predict real estate prices [40], and improve neural network training [41].

However, previous studies lack suggestions for optimizing the combinations of deep learning approaches in human posture image classification. The CNNs are deep learning methods that have been extensively studied in the field of computer vision [42–44]. One of the most important aspects that contribute to CNN's relevance in machine learning approaches is the ability to extract features from processed images automatically [45,46].

The novelty of this paper is as follows:

- A new decision support framework for the optimization of hyperparameters for AlexNet, VGG16, CNN, and multilayer perceptron (MLP) models for accomplishing optimal classification results;
- An experimental comparison of AlexNet, VGG16, CNN, and MLP classifiers that were trained and evaluated by applying the image data augmentation technique to enrich the training datasets;
- The study is the first HPO study executed on the MPII human pose dataset.

3. Materials and Proposed Method

3.1. Description of Dataset

The MPII human posture dataset comprises about 25,000 images extracted from online videos. A single image of the dataset comprises one or more persons with more than 40,000 people annotated in total. The overall dataset covers 410 human activities, and each image is provided with an activity label. Images were extracted from a YouTube video and provided with previously unannotated frames [47]. The dataset was downloaded from Kaggle as MPII Human Pose Data/Kaggle. This MPII human posture dataset asw used for training and testing the transfer learning models, which are AlexNet and VGG16. The split was a ratio of 75:25 which means 75% of the dataset for training, 15% of the dataset for

validation, and 10% for testing. The dataset used for the study was 22,000 images, broken down to 16,496 images for the training set (75%), 3305 images for validation (15%), and 2223 images for the testing (10%).

3.2. Data Preprocessing

The preprocessing procedure was performed on the MPII dataset images. The dataset was downloaded and classified into 21 classes as follows: Dancing, Home_activities, Music_playing, Occupation, Running, Sport, Winter_activities, etc. Thereafter, the dataset was normalized to size (227, 227, 3) for AlexNet, (224, 224, 3) for VGG16, (32, 32, 3) for CNN and MLP. It was rescaled to be between 0 and 255 and later changed to an array format to enable us to use it for the implementation of a deep neural network. Another major procedure in the image preprocessing stage was the use of data augmentation for the training of the MPII human posture images to generate more data for model training.

3.3. Image Data Augmentation

An image is depicted as an array of pixels that utilize grey scales or RGB values during image preprocessing. To maximize the pace of learning, the picture data should be scaled from the lowest to the highest normalization. One of the most essential image preparation approaches is image data augmentation, which may be performed both offline and online. Small datasets benefit from offline augmentation approaches, but large datasets benefit from online augmentation techniques. Because computing the model gradients using the complete huge dataset takes a long time, the data is cycled over in mini-batches during each optimization iteration with the batch size determined.

The image augmentation technique was introduced during the preprocessing procedure in this study, which generates more training data from the original data. The procedure does not require extra memory for storage, and the generated images are small batches that are discarded after the training of the model. The image augmentation techniques used are: (1) Horizontal RandomFlip; (2) RandomRotation (0.1); (3) RandomZoom (0.1); (4) Resizing (h = 32, w = 32); (5) RandomContrast (0.1)

It was discovered that the image augmentation technique introduced assisted in preventing model overfitting and enhanced the learning capacity thereby decreasing model training time complexity. The framework of the proposed classification transfer learning models with the image data augmentation method is shown in Figure 2.

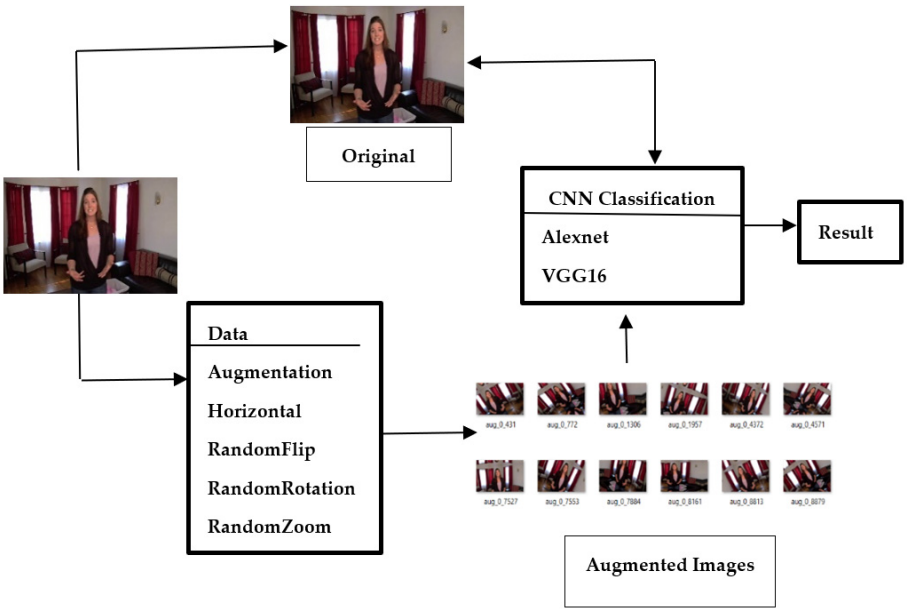


Figure 2. The framework of the proposed classification transfer learning models with the image data augmentation method.

3.4. AlexNet

The AlexNet model is characterized by containing input layers, convolutional layers, pooling layers, and fully connected (FC) layers. In general, the AlexNet transfer learning algorithm is eight layers deep that comprises 5 convolutional layers and 3 FC layers as seen in Figure 3. The network was modified by adding the Batch Normalization and Dropout layers. The sequential model was built in blocks as follows:

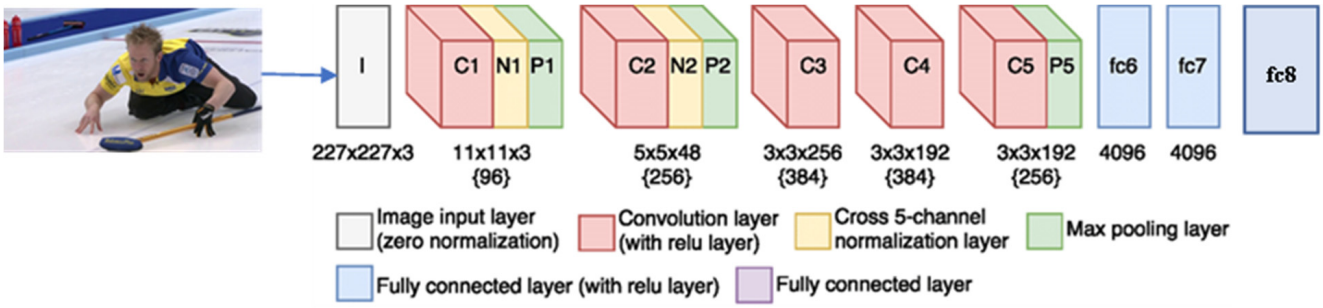


Figure 3. Modified AlexNet Architecture for Human Posture Image Classification.

Block 1

- Convolutional layer 1: 96 filters, 11 kernels, 4 strides, padding is valid, and activation = relu.
- Pooling layer 1: MaxPooling2D of size 3, 2 strides, padding is valid.
- BatchNormalization layer 1 was passed before moving to the next layer.

Block 2:

- Convolutional layer 2: 256 filters, 5 kernels, 1 stride, padding is valid, and activation = relu.
- Pooling layer 2: MaxPooling2D of size 3, 2 strides, padding is valid.
- BatchNormalization layer 2 was passed before moving to the next layer.

Block 3:

- Convolutional layer 3: 384 filters, 3 kernels, 1 stride, padding is valid, and activation = relu.
- BatchNormalization layer 3 was passed before moving to the next layer.

Block 4:

- Convolutional layer 4: 384 filters, 3 kernels, 1 stride, padding is valid, and activation = relu.
- BatchNormalization layer 4 was passed before moving to the next layer.

Block 5:

- Convolutional layer 5: 256 filters, 5 kernels, 1 stride, padding is valid, and activation = relu.
- Pooling layer 5: MaxPooling2D of size 3, 2 strides, padding is valid.
- BatchNormalization layer 5 was passed before moving to the next layer.
- The network model was flattened before moving to the next block.

Block 6:

- Fully connected layer 1: Dense node 4096, input (227, 227,3), activation = relu.
- Dropout layer was added to prevent the model from overfitting (0.4).
- BatchNormalization layer 5 was passed before moving to the next layer.

Block 7:

- Fully connected layer 1: Dense node 4096, activation = relu.
- Dropout layer was added to prevent the model from overfitting (0.4).
- BatchNormalization layer 5 was passed before moving to the next layer.

Block 8:

- Output layer: Dense (7), activation = softmax.

3.5. VGG16

The VGG16 model is characterized by containing input layers, convolutional layers, pooling layers, and FC layers as well. In general, the VGG16 transfer learning algorithm is 16 layers deep that comprises 13 convolutional layers and 3 FC layers as seen in Figure 4. It was developed into six blocks which are five convolutional blocks and one classification block. The network was modified by adding Batch Normalization and Dropout layers. The input size used was 300, and 300 and the image channel were 3.

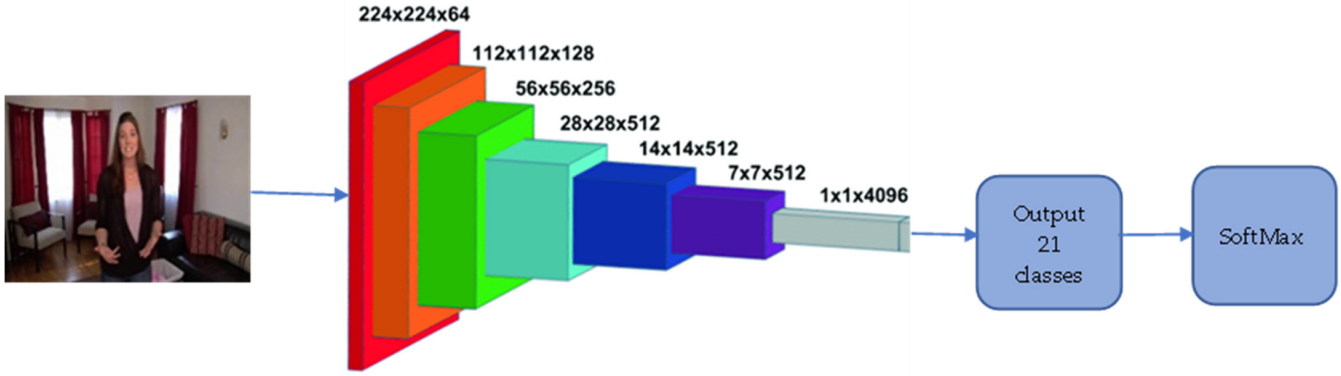


Figure 4. VGG16 Architecture for Human Posture Image Classification.

Block 1.

- Convolutional layer 1: 64 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- MaxPooling2D layer 1: Pooling size = 3, strides = 2.

Block 2.

- Convolutional layer 2: 128 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 2: 128 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- MaxPooling layer2D 2: Pooling size = 3, strides = 2.

Block 3.

- Convolutional layer 3: 256 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 3: 256 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 3: 256 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- MaxPooling2D layer 3: Pooling size 3, strides = 2.

Block 4.

- Convolutional layer 4: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 4: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 4: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- MaxPooling2D layer 4: Pooling size = 3, strides = 2.

Block 5.

- Convolutional layer 5: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- Convolutional layer 5: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.

- Convolutional layer 5: 512 filters, 3 kernel, activation = relu, padding = same, image_input as defined earlier.
- MaxPooling2D layer 5: Pooling size = 3, strides = 2.

Block 6 (Classification Block)

- Flatten.
- FC layer 1: Dense = 4096 nodes, activation = relu.
- Dropout = 0.4.
- FC layer 2: Dense = 4096 nodes, activation = relu.
- Dropout = 0.4.
- Output layer: classes, activation = softmax.

3.6. Proposed System Architecture

In this study, we used the RandomSearch hyperparameter optimization approach to optimize two CNN transfer learning algorithms for decision support, namely AlexNet and VGG16. These algorithms are employed in MPII human posture analysis to classify human poses. These datasets consist of around 25,000 poses of different activities. When the default hyperparameter settings are used, the accuracy of the algorithms is determined, and then, when each of the HPO techniques is used, it is computed again. There is a comparison between the before and after photos. For enriching the initially limited dataset, we also design an image data augmentation approach. Rotation, translation, zoom, flips, shear, mirror, and color perturbation [48] are examples of picture data augmentation techniques that solve the problem of insufficient training data by including altered original samples in the training set (Figure 2). The classification results with image data augmentation were verified based on AlexNet [49] and VGG16 [5], respectively.

The proposed system architecture is presented in Figure 5. This represents the classification of human postures using data augmentation at the preprocessing phase after which the models were trained. The MPII dataset was supplied to the system, and the preprocessing phase, which includes normalization, rescaling, and data augmentation, was conducted after which the processed dataset was passed to the models for training. The results were obtained, and the system was evaluated using the training and validation losses and accuracies. Figure 6 shows the proposed system using the hyperparameter-optimization method. The MPII dataset was supplied into the system, and the preprocessing such as normalization and rescaling was conducted. The hyperparameter optimizer was used on the processed data, after which it was passed to the models for training. The system was evaluated using training and validation losses and accuracies.

The proposed approach relied on image data augmentation, hyperparameters, transfer learning techniques, and classification approaches that were utilized on human posture datasets. The recognition of human postures has become a vital aspect of the scientific area in current centuries.

3.7. Hyperparameter Optimization (HPO)

The structural organization and learning techniques of neural networks are governed by a set of hyperparameters known as structural and algorithm hyperparameters [50]. The structure and topology are characterized by structural hyperparameters, which include the number of network layers, the number of neurons in each layer, the degree of connection, the transfer function, etc. As the network's structure changes, they have an impact on its effectiveness and computational complexity. The size of the training dataset, the training method, momentum, learning rate, and other algorithm parameters all affect the learning process. Although hyperparameters are not part of the neural network model and have no impact on its performance, they do have an impact on the training stage's speed and performance.

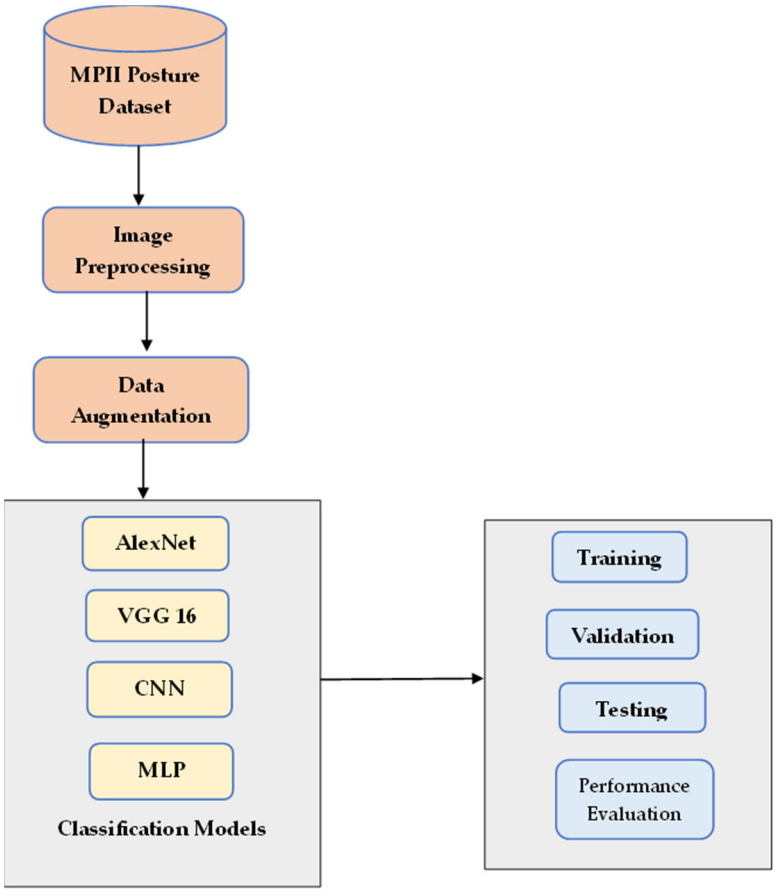


Figure 5. Proposed Image Data augmentation with the model's architecture.

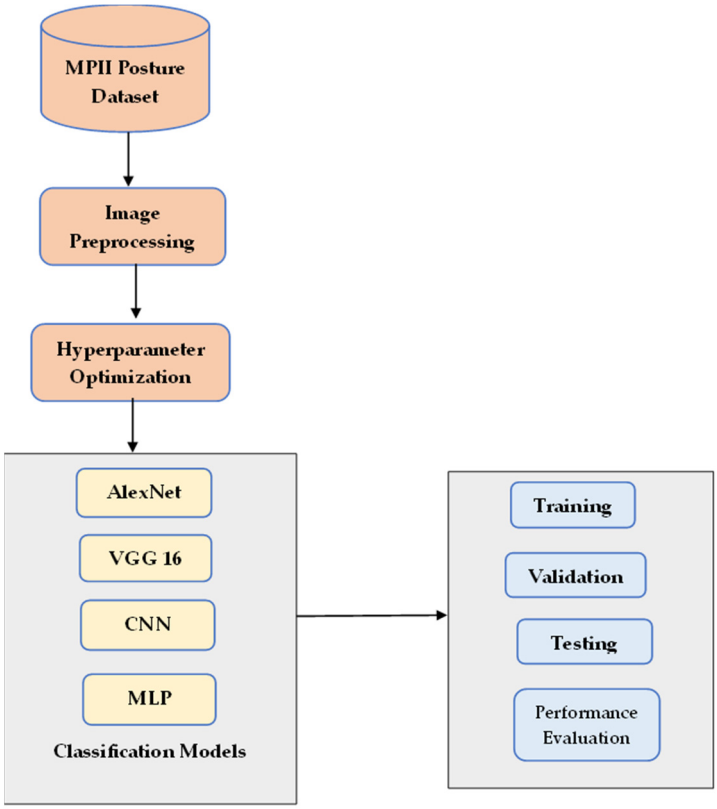


Figure 6. Proposed decision support models' hyperparameter-optimizer architecture.

The HPO settings for machine learning or deep learning algorithms are a collection of resolutions that have a true impact on the training procedure and the classification results, which reflect how a model performs appropriately. The procedure of training a model to classify human pose images in the training dataset and apprehend the output based on the image patterns is referred to as model training. Aside from hyperparameter selection, model design, which describes a model, directly influences how long it takes to train, validate, and test a model. The setting has arisen as an important and problematic subject in the application of deep learning algorithms because of their impact on model performance and the fact that the ideal collection of values is unknown. Hyperparameters can be adjusted in a variety of methods in the literature. The procedures for optimizing these hyperparameters are presented as follows:

- When the scholar has a firm understanding of neural network structure and learning data, the manual search calculates the hyperparameter value based on the scholar's perception or knowledge. However, the criteria for setting hyperparameters are ambiguous, demanding multiple experiments.
- Random search is a method to train a model that selects random combinations of hyperparameters. We utilize the best combinations of random hyperparameters. Random search resembles grid search in several ways.

The fact that we do not give a list of feasible values for each hyperparameter is a critical distinction. Instead, for each hyperparameter, we sample values from a statistical distribution. For each hyperparameter, a sample distribution is constructed to perform a random search. This method allows us to limit the number of hyperparameter combinations that are attempted. In contrast to grid search, which attempts every conceivable combination, random search allows us to define the number of models to train. Our search iterations might be based on our computational resources or the time spent per iteration.

An experiment is a set of tests designed to identify the factors that have the greatest impact on a response variable [51]. The major goal of the Design-of-Experiments (DOE) methodology is to maximize this response variable after these components are found. To discover the link between factors and the response variable, these studies need a careful selection of variables, their ranges, and the number of experiments run. The influence of factors on the response variable has traditionally been examined by changing the amounts of one component at a time while keeping the other factors constant. However, this method is inefficient and leaves out information about prospective interactions.

Table 1 provides an overview of all hyperparameters that were fine-tuned for all models. The most significant parameters of neural networks are the initial learning rate, the learning rate decay factor, the number of hidden neurons, and regularization strength.

Table 1. Hyperparameters and their range for the four models.

Hyperparameter	Description	AlexNet Range	VGG16 Range	CNN Range	MLP Range
Kernel Size	Convolutional layer kernel size	3, 5, 7, 11	3, 5, 7, 11	3, 5, 7, 11	-
Filter number	Convolutional layer neurons	Min_value = 64, 128, 256 Max_value = 256, 384, 512	Min_value = 64, 128, 256 Max_value = 256, 384, 512	Min_value = 32, 64 Max_value = 256, 384	-
Layer Depth	Number of layers in the network	1, 2, 3	1, 2, 3	1, 2	-
Neuron Count	Final FC layers neuron count	Min_value = 1028 Max_value = 4500	Min_value = 1028, 1028 Max_value = 4500, 4500	Min_value = 28 Max_value = 256	Min_value = 128, 256 Max_value = 512, 1028
Learning Rate	Updating weight while training	10^{-2}, 10^{-3}, 10^{-4}	10^{-2}, 10^{-3}, 10^{-4}	10^{-2}, 10^{-3}, 10^{-4}	10^{-2}, 10^{-3}, 10^{-4}

In the validation dataset, the best-performing model with the minimum loss value is chosen. The ultimate performance of the model is estimated using a hold-out test set, as the validation dataset's performance is integrated into the model's hyperparameter optimization. This strategy provides an objective assessment of performance. Tables 2–5 show the best hyperparameter results for AlexNet, VGG16, CNN, and MLP, respectively.

Table 2. Summary of AlexNet Models hyperparameter search result (Best Trials).

Hyperparameter	Values
Conv_1_filter	232
Conv_1_kernel	5
Conv_2_filter	352
Conv_2_kernel	7
Conv_3_filter	312
Conv_3_kernel	7
Dense_1_units	1540
Dense_2_units	2564
Learning_rate	0.0001
Score	0.8910

Table 3. Summary of VGG16 Models hyperparameter search result (Best Trials).

Hyperparameter	Values
Conv_1_filter	232
Conv_1_kernel	3
Conv_2_filter	128
Conv_2_kernel	7
Conv_3_filter	424
Conv_3_kernel	11
Dense_1_units	2052
Dense_2_units	3076
Learning_rate	0.001
Score	0.9022

Table 4. Summary of CNN Models hyperparameter search result (Best Trials).

Hyperparameter	Values
Conv_1_filter	32
Conv_1_kernel	5
Conv_2_filter	64
Conv_2_kernel	7
Dense_1_units	60
Learning_rate	0.01
Score	0. 8742

Table 5. Summary of MLP model hyperparameter search result (Best Trials).

Hyperparameter	Values
Dense_1_units	512
Dense_2_units	640
Learning_rate	0.0001
Score	0.8685

4. Results and Discussion

This section presents the implementation results for all the models used in the study. The models that were executed alone, with data augmentation, and with hyperparameter optimization are presented in figures and tables in this section.

4.1. Experimental Settings

Training and validation accuracies and losses are used to verify the models' performance. The entire execution results are displayed in tables and graphs for easier comprehension. The entire experiments were executed on an Intel (R) Core (TM) i7-6600U CPU running at 2.60 GHz with 8 GB of RAM and a 64-bit operating system.

4.2. Optimization of Hyperparameter of the Transfer Learning Models

4.2.1. AlexNet

The AlexNet transfer learning algorithm was implemented alone (Figure 7a,b) without augmenting the image dataset while conducting the preprocessing phase. The dataset was only normalized and resized. The parameters set for the implementation are 50 epochs, batch size of 32, image input shape of (227, 227), channel = 3, optimizer = Adam, loss is categorical_crossentropy, and learning rate = 0.0001. The algorithm was implemented with IDA, and the results are shown in Figure 8a,b. The result of the implementation of the algorithm with a hyperparameter optimizer is shown in Figure 9a and b. From the implementation, it was discovered that the AlexNet + HPO model performed the best in terms of training accuracy at 99.9%, while AlexNet + IDA performed the second best with a training accuracy of 98.8%. The validation accuracy of 91.2% for AlexNet + HPO was the best with the lowest execution time of 28 min. Table 6 shows the summary of all AlexNet results.

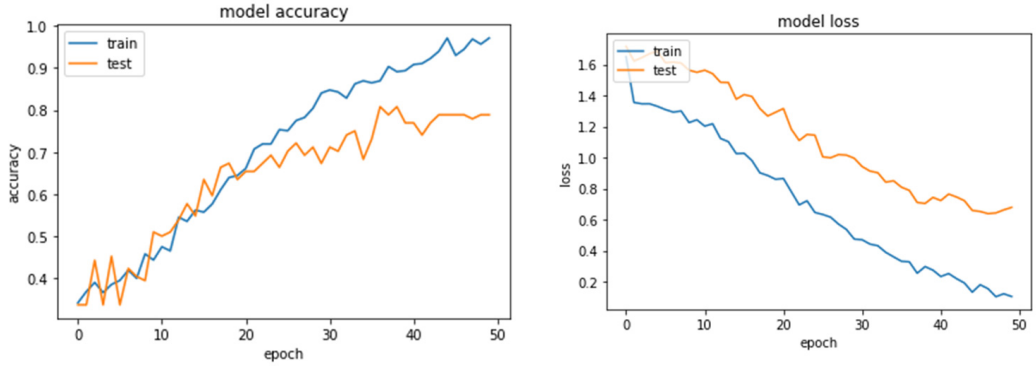


Figure 7. Training and testing of AlexNet model: (**a**) accuracy, (**b**) loss.

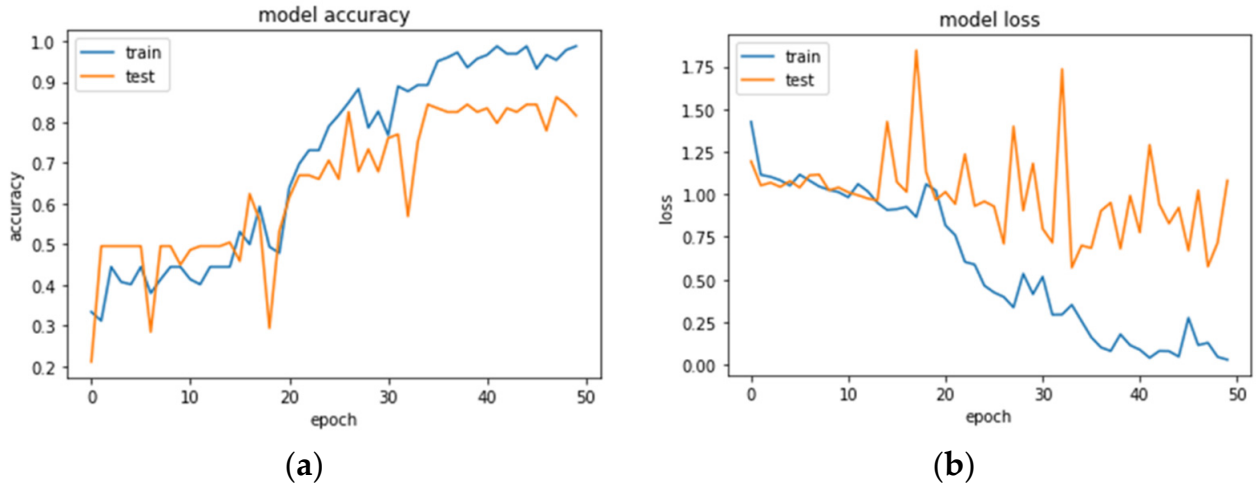


Figure 8. Training and testing of AlexNet + IDA model: (**a**) accuracy, (**b**) loss.

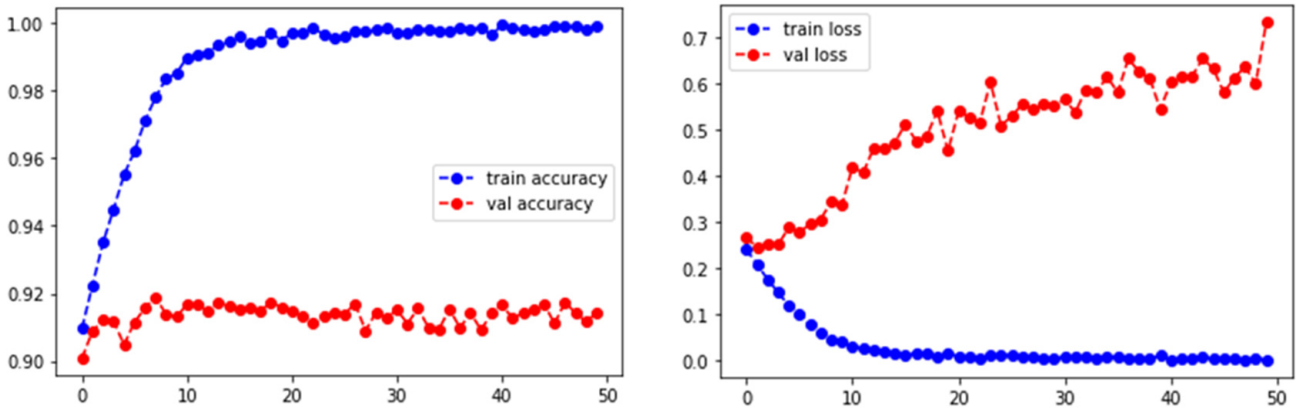


Figure 9. Training and testing of AlexNet + HPO model: (**a**) training and validation accuracy, (**b**) training and validation loss.

Table 6. Summary of AlexNet transfer learning results.

Transfer Learning Model	Training Accuracy	Training Loss	Testing Accuracy	Validation Loss	Time (min)
AlexNet	0.9709 = 97.1%	0.1050	0.7885 = 78.9%	0.6788	61
AlexNet + IDA	0.9877 = 98.8%	0.0293	0.8165 = 81.7%	1.0808	50
AlexNet + HPO	0.9990 = 99.9%	0.0043	0.9147 = 91.2%	1.0356	28

4.2.2. VGG16 Transfer Learning Model

The VGG16 transfer learning algorithm was implemented alone (Figure 10a,b) without augmenting the image datasets while conducting the preprocessing phase. The dataset was only normalized and resized. The parameters set for the implementation are epochs of 50, batch size of 32, image input shape of (300, 300), channel = 3, optimizer = Adam, loss of categorical_crossentropy, and learning rate of 0.0001. The algorithm was also implemented with IDA, and the results are shown in Figure 11a,b. The result for the implementation of the algorithm with a hyperparameter optimizer is shown in Figure 12a,b. From the implementation, it was discovered that the VGG16 model performed best in terms of a training accuracy of 100%, while VGG16 + HPO performed second best with a training accuracy of 99.8%. The validation accuracy of 90.2% for VGG16 + HPO was the best with the lowest execution time of 33 min. Table 7 shows the summary of the entire VGG16 result executions.

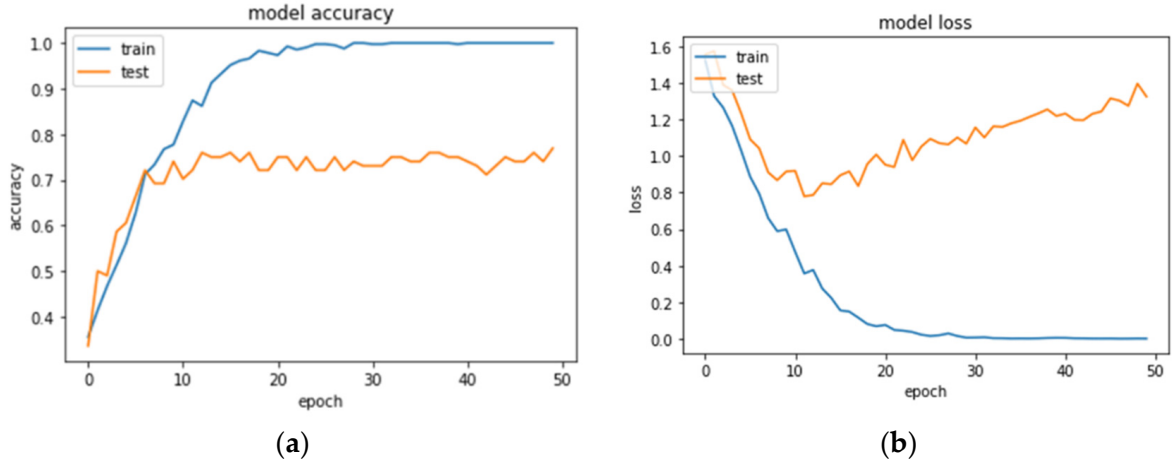
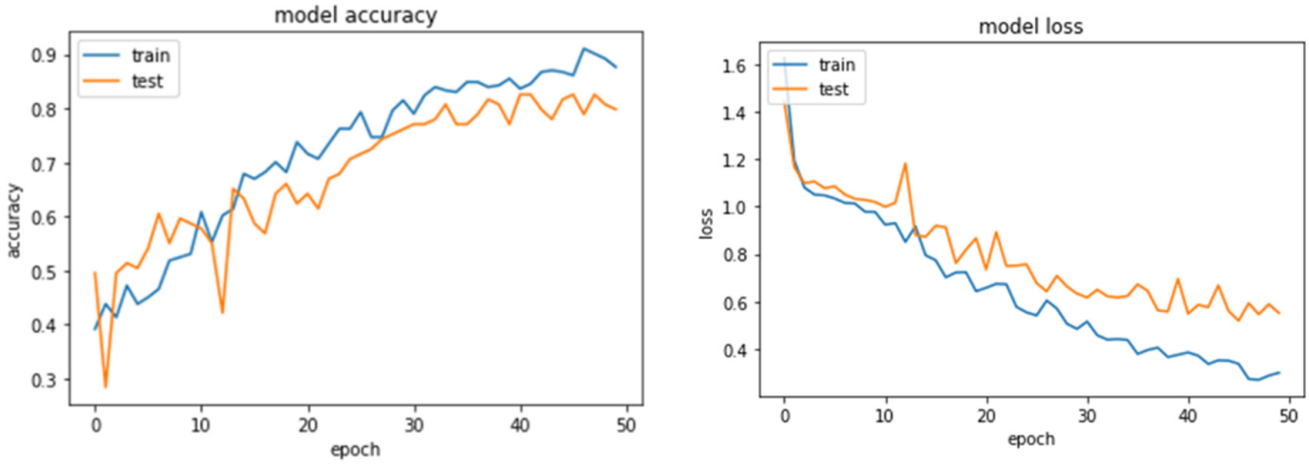
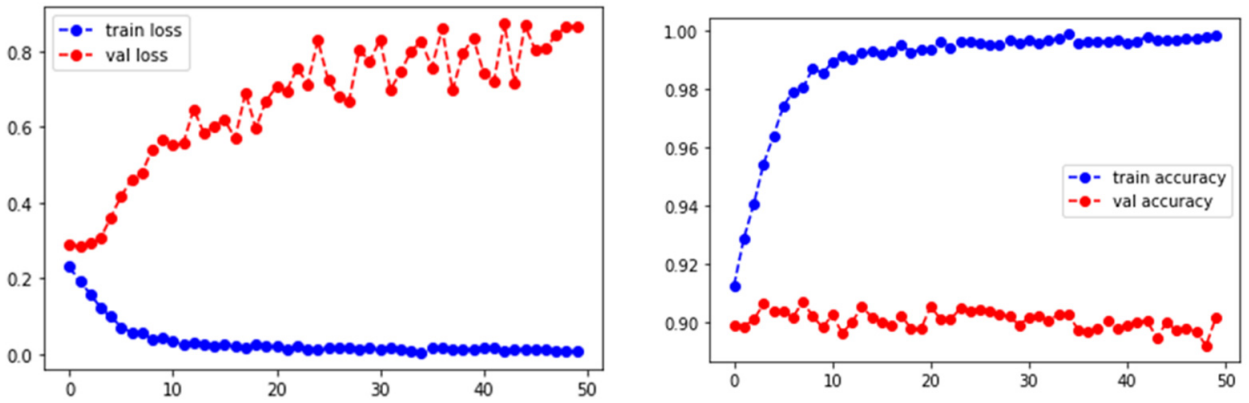


Figure 10. Training and testing of VGG16 model: (**a**) training and testing accuracy, (**b**) training and testing loss.



(**a**) (**b**)

Figure 11. Training and testing of VGG16 + IDA model: (**a**) training and testing accuracy, (**b**) training and testing loss.



(**a**) (**b**)

Figure 12. Training and testing of VGG16 + HPO model: (**a**) training and validation accuracy, (**b**) training and validation loss.

Table 7. Summary of VGG16 transfer learning results.

Transfer Learning Model	Training Acc.	Training Loss	Testing Acc.	Validation Loss	Time (min)
VGG16	0.1000 = 100%	0.0011	0.7692 = 76.9%	1.3228	83
VGG16 + IDA	0.8765 = 87.7%	0.3005	0.7982 = 79.8%	0.5521	78
VGG16 + HPO	0.9984 = 99.8%	0.0067	0.9015 = 90.2%	0.8654	33

4.3. Standard Classifier Results Examination

The performance of each deep learning (CNN and MLP) classifier with data augmentation, hyperparameter optimization, and alone was measured and is detailed in Tables 8 and 9 for CNN and MLP, respectively.

Table 8. Summary of CNN results.

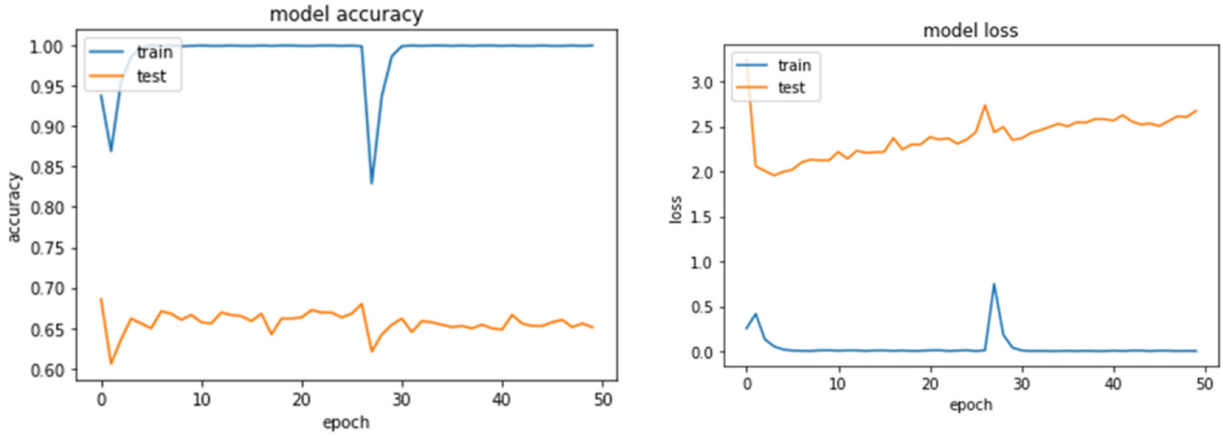
Transfer Learning Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Time (min)
CNN	0.9996 = 99.9%	0.0040	0.6511 = 65.1%	2.6715	39
CNN + IDA	0.6873 = 68.7%	0.8018	0.6767 = 67.7%	1.3748	31
CNN + HPO	0.9869 = 98.7%	0.0530	0.8750 = 87.5%	2.1469	3

Table 9. Summary of MLP results.

Transfer Learning Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Time
MLP	0.9928 = 99.3%	0.0386	0.7009 = 70.1%	2.0471	38 min
MLP + IDA	0.6930 = 69.3%	0.8584	0.7205 = 72.1%	0.8379	33 min
MLP + HPO	0.9746 = 97.5%	0.0774	0.8985 = 89.9%	0.3604	7 min

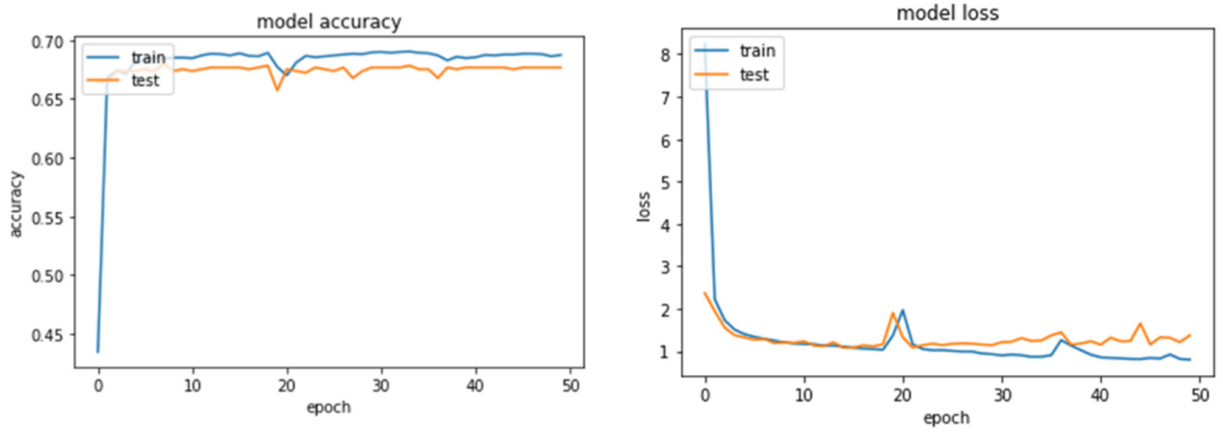
4.3.1. CNN Model

The CNN model was implemented alone (Figure 13a,b) without augmenting the image datasets while conducting the preprocessing phase. The dataset was only normalized and resized. The parameters set for the implementation are epochs of 50, batch size of 32, image input shape of (32, 32), channel = 3, optimizer = Adam, loss is categorical_crossentropy, and learning rate = 0.0001. The algorithm was also implemented with IDA, and the results are shown in Figure 14. The result for the implementation of the algorithm with hyperparameter optimization is shown in Figure 15. From the implementation, it was discovered that the CNN model performed best in terms of a training accuracy of 99.9%, while CNN + HPO performed second best with a training accuracy of 98.7%. The validation accuracy of 87.5% for CNN + HPO was the best with the lowest execution time of 3 min. Table 8 shows the summary of the CNN results.



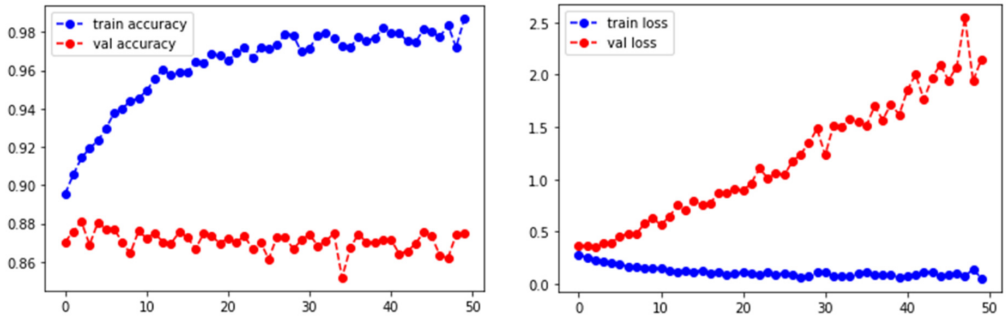
(**a**) (**b**)

Figure 13. Training and testing of CNN model: (**a**) training and testing accuracy, (**b**) training and testing loss.



(**a**) (**b**)

Figure 14. Training and testing of CNN + DA model: (**a**) training and testing accuracy, (**b**) training and testing loss.

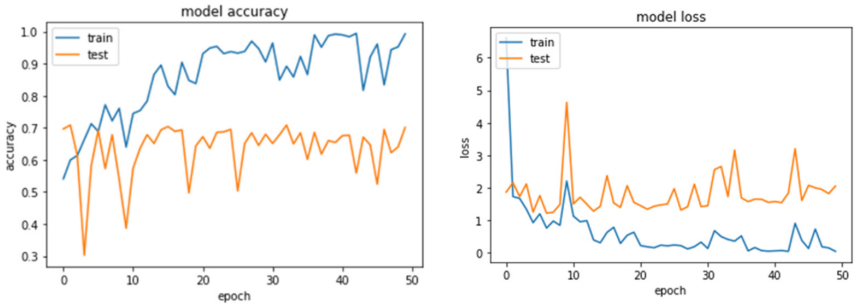


(**a**) (**b**)

Figure 15. Training and testing of CNN + HPO model: (**a**) training and validation accuracy, (**b**) training and validation loss.

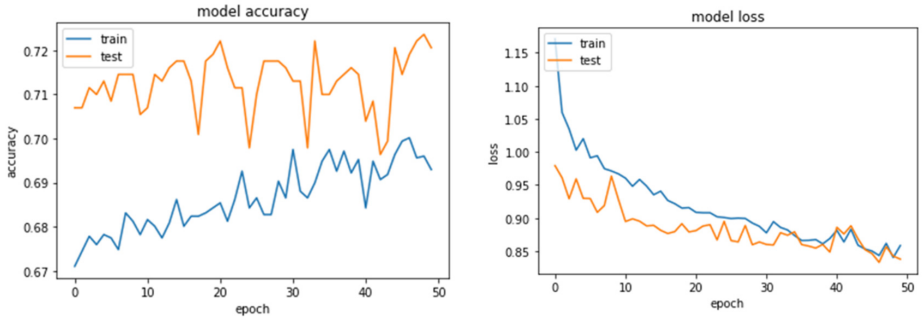
4.3.2. MLP Model

Figures 16–18 give the MLP classifier alone result, with image augmentation, and with the hyperparameter optimizer, respectively. The MLP algorithm was implemented alone (Figure 16a,b) without augmenting the image datasets while conducting the preprocessing phase. The dataset was only normalized and resized. The parameters set for the implementation are epochs of 50, batch size of 32, image input shape of (32, 32), channel = 3, optimizer = Adam, loss of categorical_crossentropy, and learning rate of 0.0001. The algorithm was also implemented with IDA, and the results are shown in Figure 17a,b. The result for the implementation of the model with hyperparameters is shown in Figure 18a,b. From the implementation, it was discovered that the MLP model performed best in terms of training accuracy at 99.3%, while MLP + HPO performed second best with a training accuracy of 97.5%. The validation accuracy of 89.9% for CNN + HPO was the best with the lowest execution time of 7 min. Table 9 shows the summary of the entire MLP result executions.



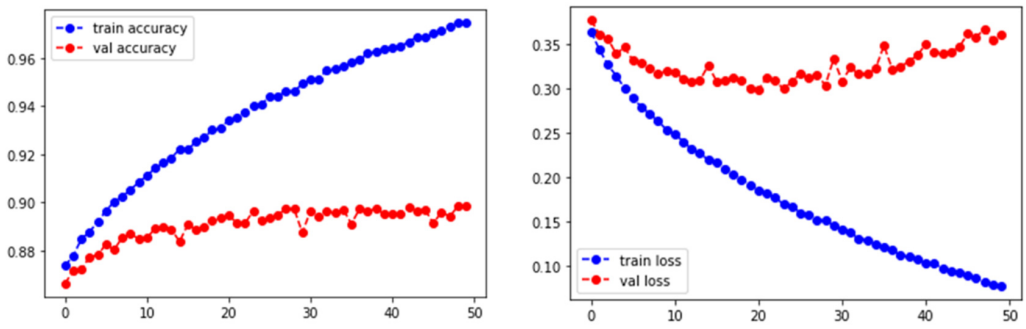
(**a**) (**b**)

Figure 16. Training and testing of MLP model: (**a**) training and testing accuracy, (**b**) training and testing loss.



(**a**) (**b**)

Figure 17. Training and testing of MLP + DA model: (**a**) training and testing accuracy, (**b**) training and testing loss.



(**a**) (**b**)

Figure 18. Training and testing of MLP + HPO model: (**a**) training and validation accuracy, (**b**) training and validation loss.

4.4. Performance Analysis of the Models

The proposed models were evaluated using training and validation accuracies and losses, which are shown in Figure 19, while Figure 20 shows the training and validation losses of the models. VGG16 has the best training accuracy with the lowest training loss of 0.0011, while AlexNet + HPO has the second-best training accuracy of 99.9% with a training loss of 0.0043, and VGG 19 + HPO is the third-best with a training accuracy of 99.8% and a training loss of 0.0067. AlexNet + HPO performed best with a validation accuracy of 91.2%, and VGG16 + HPO had a VA of 90.2%, which made it the second-best model. The model with the lowest validation loss (0.5521) is VGG16 + IDA followed by AlexNet with a validation loss of 0.6788.

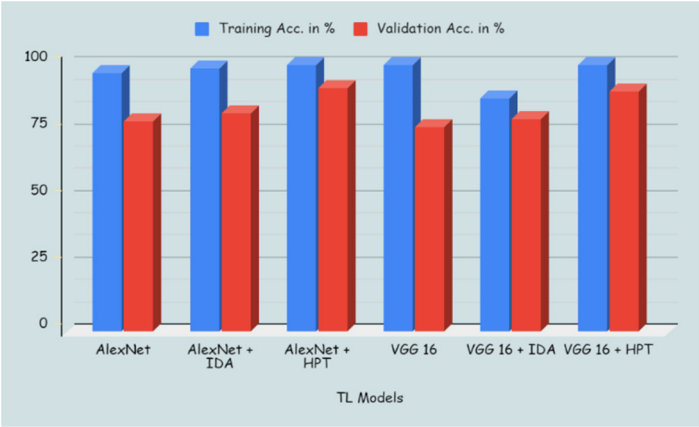


Figure 19. Transfer learning models' training and validation accuracies.

Figure 20. Transfer learning models' training and validation losses.

The training and validation accuracies of the transfer learning models are shown in Figure 21, while Figure 22 shows the training and validation losses of the models. CNN has the best training accuracy of 99.9% with the lowest training loss of 0.0040, while MLP has the second-best training accuracy of 99.3% with a training loss of 0.0386, and CNN + HPO is the third-best with a training accuracy of 98.7% and a training loss of 0.0530. MLP + HPO performed best with a validation accuracy of 89.9%, and CNN + HPO had a validation accuracy of 87.5%, which made it the second-best model. The model with the lowest validation loss (0.3604) is MLP + HPO, followed by MLP + IDA with a validation loss of 0.8379.

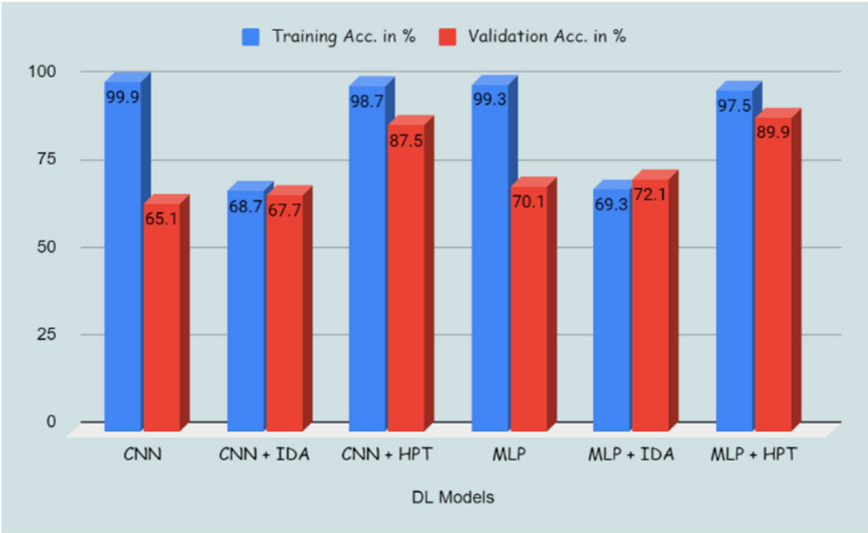


Figure 21. Deep learning models' training and validation accuracies.

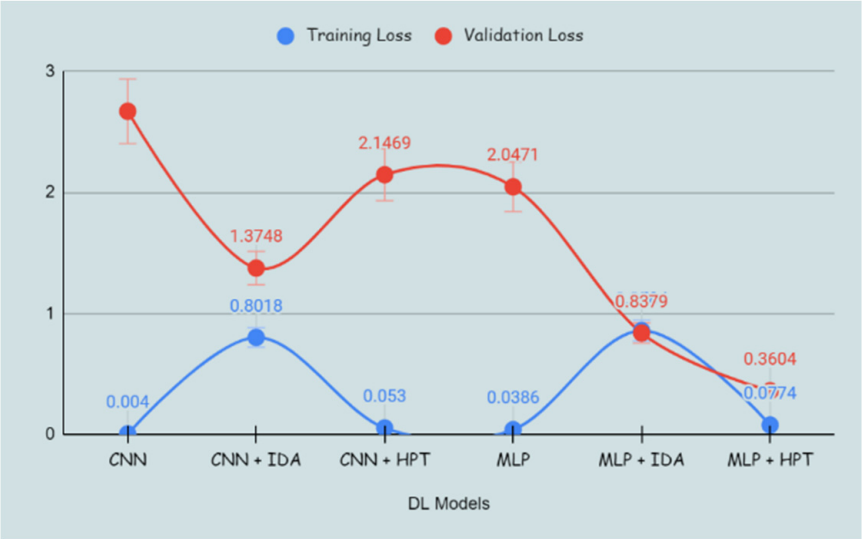


Figure 22. Deep learning models' training and validation losses.

The time taken for execution was also used for the evaluation of the models. It was discovered that those transfer learning algorithms that used HPO were executed at the lowest time. AlexNet + HPO executed at 28 min, while VGG16 + HPO executed at 33 min. It was also discovered that the next models with a lower time of execution are the ones that used image data augmentation at the preprocessing stage of their datasets. AlexNet + IDA executed at 50 min, while VGG16 + IDA executed at 78 min as seen in Table 10.

Table 10. Transfer learning models with execution time.

Transfer Learning Models	Time
AlexNet	61 min
AlexNet + IDA	50 min
AlexNet + HPO	28 min
VGG16	83 min
VGG16 + IDA	78 min
VGG16 + HPO	33 min

The same condition applied to deep learning models have the models implemented with HPO having the lowest time of execution. The deep learning models with a second-lowest time of execution were also the ones that used IDA at the preprocessing stage of the datasets. CNN + HPO executed in 3 min, while MLP + HPO executed in 7 min. Likewise, CNN + IDA was executed in 31 min, and MLP + IDA was executed in 33 min (Table 11).

Table 11. Deep learning models with execution time.

Deep Learning Models	Time
CNN	39 min
CNN + IDA	31 min
CNN + HPO	3 min
MLP	38 min
MLP + IDA	33 min
MLP + HPO	7 min

4.5. Comparison with Related Work

Several previous studies proposed methods for posture recognition while using the MPII dataset for performance evaluation.

Luvizon et al. [52] suggested a deep network architecture for pose recognition. Input images are processed through a series of CNNs made up of prediction blocks (PB), downscaling and upscaling units (DU and UU), and simple (skip) connections; the entry-flow derives feature maps from images. Each PB generates supervised posture and action predictions, which are then honed by more units and blocks. Pose estimation and action recognition information is transferred separately from one prediction block to another.

Munea et al. [53] outlined a quick and effective bottom-up multi-person posture estimate method based on a few deconvolutional layers applied to a ResNet architecture. These bottom-up detection and association representations are simultaneously inferred to encapsulate the global context for a greedy parse to produce good results at a cheap cost.

In Qin et al. [54], a simple Human Pose Estimation network with RGB picture input was presented. They proposed the Capable and Vigorous Campstool Network (CVC-Net) using the Stacked Hourglass network design. They suggested a new residual block called the Res2Net depth block and used it to replace the residual blocks in the Hourglass network to decrease the number of model parameters. They employed three methods—the channel attention mechanism, the PixelShuffle up-sampling approach, and a proposed Cross-Stage Heatmap Fusion method—to enhance model performance. They used a Differentiable Spatial to Numerical Transform model together with Euclidean distance loss in the coordinate regression phase so that the model could be trained end-to-end.

Wang et al. [55] improved performance by combining the benefits of Transformers and high resolution. To be more precise, we create MTNet (Multi-scale Transformers-based high-resolution Networks), a sub-network with two parallel branches. One is the local branch and combines high resolution with local convolutional procedures. Another uses multi-scale Transformer encoders to learn long-range interdependence of full body keypoints. Finally, both two branches are combined to forecast final keypoint heatmaps.

Wang et al. [56] created a re-parameterized lightweight bottleneck block UULPN that includes a lot of feature maps and broadens their variety. In the bottleneck block, they

offered a multi-branch structure and a single-branch structure. A multi-branch structure was used in the training phase to improve prediction accuracy. A single-branch structure was employed in the deployment phase to increase the model inference speed.

In Wu et al. [57], to estimate a 3D human posture, a mixture density network that predicts a variety of potential hypotheses was proposed. A feature extractor initially estimates the 2D joint points from the input photos before extracting the correlation data for human joints. Following the extraction of the human posture characteristic, the hypotheses generator generates pose hypotheses. Moreover, they presented the Locally Connected Network (LCN) instead of the conventional Fully Connected Network (FCN), which is applied to a feature extraction module, to use the link between human joints better. The projected pose is then scored using a 3D pose picker based on the ordinal ranking of joints.

Yang et al. [58] suggested adding a shortcut link to a dense layer, which was inspired by the architecture of a residual block in ResNet. To increase the network's convergence, the number of parameters is significantly reduced, and the computational cost is also decreased when all residual blocks are replaced with a densely connected residual module.

Zhang et al. [59] described a framework for estimating human position. They adopted the MobileNetV2 backbone network design for posture estimation and minimized computational cost with hardly any accuracy loss by employing a differentiable neural architecture search approach.

The performance of previous studies is summarized in Table 12 and compared against our method.

Table 12. Comparison with previous studies on posture recognition using MPII dataset.

Reference	Model	Accuracy
Luvizon et al. [52]	Custom network	87.0%
Munea et al. [53]	Modified ResNet	93.5%
Qin et al. [54]	CVC-Net	91.6%
Wang et al. [55]	Transformer network	90.86%
Wang et al. [56]	UULPN	85.7%
Wu et al. [57]	Mixture density network	Only qualitative results reported
Yang et al. [58]	Modified ResNet	88.8%
Zhang et al. [59]	MobileNetV2 backbone	88.1%
This study	**MLP + HPO**	**89.9%**

5. Discussion

The results of hyperparameter optimization reveal that some combinations of parameters have a greater impact on the model's performance, while others have a minor impact. We noticed that the number of layers and the breadth of the filter had a significant impact on the prediction performance. The results also showed that great performance may be achieved for all filter widths. Furthermore, using several layers produced somewhat better performance than using only one layer since it tolerates additional model complexity, but it similarly required a longer training time. The training time was affected by the number of layers and the breadth of the filter, but not by the classification performance. Accordingly, provided that the number of layers is rigid, a high filter width takes fewer periods to train than a lesser filter width, even though the two alternatives provide similar predictions. The same can be said for the number of filters used. The higher the filters used, the longer the training period becomes, with no discernible gain in prediction accuracy. The number of filters used has a significant impact on the training time (bottom). The number of trainable parameters is determined by the number of filters, their breadth, and the number of layers and stacks. As a result, by utilizing fewer filters for the equivalent fuse of filter width and layers, training time is lowered. Adding extra layers to the network enhances the depth and, as a result, its complexity. Training time is influenced by the number of layers but not by performance. Although both would deliver equal outcomes provided the number of layers was fixed, an extensive filter width would require fewer layers and hereafter fewer training times than a narrower filter width.

An independent test set was used to choose the most efficient network model with the optimal mix of hyperparameter values. These results show that the suggested VGG16 and AlexNet transfer learning models can reliably classify human pose images. As a result, transfer learning models are an excellent choice for replacing time-consuming classical machine learning models. Figures 9–19 depict the model training outcomes, including training accuracies, validation accuracies, training loss, and validation loss numbers. The fundamental downside of transfer learning models is that they demand a lot of processing resources, such as GPUs and a lot of RAMs. Transfer learning classifiers create synthetic data during the training phase, which requires a large amount of storage space.

Image augmentation was used in the data preparation portion of the investigation. Data augmentation was used as a regularizer to help manage data overfitting. By producing additional training data and exposing the model to diverse versions of data, image augmentation helps to reduce the likelihood of overfitting. The image augmentation utilized in this study helped the model to operate better and more precisely by improving the results. It also reduces operating expenses by adding transformations in the datasets, as evidenced by the fact that the model calculation execution time is shorter than when the model is without image augmentation. It also helps with data cleansing, which is necessary for good model accuracy. Image augmentation also improves the robustness of deep learning by adding variations to the model.

Deep learning models are particularly subtle to the extent of the training set and, to be adequately built, require significantly bigger training datasets. Our findings reveal that no single hyperparameter combination significantly beats the others. Training a model with the same hyperparameter settings again does not necessarily result in the same classification accuracy due to variations in weight and bias initialization. It is critical to go through the training process many times before deciding on the best-performing network. Deeper models with more layers, on the other hand, take longer to train.

The proposed hyperparameter-optimization methodology for decision-making can support doctors in making critical clinical decisions more effectively. The approach presented in this paper is also useful in situations where people lack access to integrated primary medical care technology for early diagnosis and treatment.

6. Conclusions

This study used four models for decision support in posture recognition: two transfer learning algorithms and two deep learning algorithms CNN and MLP. The models were implemented on MPII Human-Posture dataset images. Three main stages were carried out, which were implementing the algorithms alone, implementing using image augmentation, and implementing using hyperparameter optimization (HPO). The HPO transfer learning algorithms outperformed the ones implemented with image augmentation in terms of training loss and validation accuracy. AlextNet + HPO outperformed the other four models with a validation accuracy of 91.2% followed by VGG16 + HPO with a validation accuracy of 90.2%. The algorithm with the lowest training loss was VGG16 (0.0011), while the model with the lowest validation loss was 0.5521. In terms of execution time, deep learning models with HPO had the lowest execution times of 3 min for CNN + HPO and 7 min for MLP + HPO. This was a result of having fewer layers. Therefore, we recommend that researchers implement their transfer learning algorithms using hyperparameter-optimization techniques to obtain optimized training and validation losses and accuracies.

In image classification, particularly using transfer learning and deep learning models, image augmentation can generate diverse outcomes based on a different dataset. In this study, the performance of posture image classification was determined using transfer learning models. In general, the proposed models are effective in their decision-making application for classifying the MPII pose dataset, as evidenced by the comparison of the four models.

The disadvantage is the increased complexity, as finding optimal hyperparameter values requires additional computational resources. The scope of this study will be ex-

panded in the future by performing experiments with larger image datasets. This may be accomplished by integrating deep learning algorithms with optimization methods to improve image data augmentation and accurately characterize postures.

Author Contributions: Conceptualization, R.M.; methodology, R.M.; software, R.O.O.; validation, R.O.O., R.M. and R.D.; formal analysis, R.O.O., R.M. and R.D.; investigation, R.O.O. and R.M.; resources, R.M.; data curation, R.O.O.; writing—original draft preparation, R.O.O. and R.M.; writing—review and editing, R.D.; visualization, R.O.O. and R.M.; supervision, R.M.; funding acquisition, R.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The codes for the implementation of this study are available upon request.

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

Abbreviations

CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
GPU	Graphics Processing Unit
HPO	Hyperparameter Optimization
DOE	Design of Experiments

References

1. Deng, L. An overview of deep-structured learning for information processing. In Proceedings of the Asia-Pacific Signal and Information Processing Annual Summit Conference (APSIPA-ASC), Xi'an, China, 19–21 October 2011; pp. 1–14.
2. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
3. 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*, 211–252. [CrossRef]
4. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *60*, 1–25.
5. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
6. 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 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
7. Zhang, M.; Jing, W.; Lin, J.; Fang, N.; Wei, W.; Woźniak, M.; Damaševičius, R. NAS-HRIS: Automatic design and architecture search of neural network for semantic segmentation in remote sensing images. *Sensors* **2020**, *20*, 1–15. [CrossRef] [PubMed]
8. Ogundokun, R.O.; Maskeliunas, R.; Misra, S.; Damaševičius, R. Improved CNN Based on Batch Normal-ization and Adam Optimizer. In Proceedings of the International Conference on Computational Science and Its Applications, Malaga, Spain, 4–7 July 2022; Springer: Cham, Switzerland, 2022; pp. 593–604.
9. Ben Gamra, M.; Akhloufi, M.A. A review of deep learning techniques for 2D and 3D human pose estimation. *Image Vis. Comput.* **2021**, *114*, 104282. [CrossRef]
10. Song, L.; Yu, G.; Yuan, J.; Liu, Z. Human pose estimation and its application to action recognition: A survey. *J. Vis. Commun. Image Represent.* **2021**, *76*, 103055. [CrossRef]
11. Jayatilake, S.M.D.A.C.; Ganegoda, G.U. Involvement of machine learning tools in healthcare decision making. *J. Healthc. Eng.* **2021**, *2021*, 6679512. [CrossRef] [PubMed]
12. de Siqueira, V.S.; Borges, M.M.; Furtado, R.G.; Dourado, C.N.; da Costa, R.M. Artificial intelligence applied to support medical decisions for the automatic analysis of echocardiogram images: A systematic review. *Artif. Intell. Med.* **2021**, *120*, 102165. [CrossRef]
13. Tsougos, I.; Vamvakas, A.; Kappas, C.; Fezoulidis, I.; Vassiou, K. Application of radiomics and decision support systems for breast MR differential diagnosis. *Comput. Math. Methods Med.* **2018**, *2018*, 7417126. [CrossRef] [PubMed]
14. Yang, Y.; Feng, X.; Chi, W.; Li, Z.; Duan, W.; Liu, H.; Liang, W.; Wang, W.; Chen, P.; He, J.; et al. Deep learning aided decision support for pulmonary nodules diagnosing: A review. *J. Thorac. Dis.* **2018**, *10*, S867–S875. [CrossRef]
15. Ali, L.; Rahman, A.; Khan, A.; Zhou, M.; Javeed, A.; Khan, J.A. An automated diagnostic system for heart disease prediction based on $\chi 2$ statistical model and optimally configured deep neural network. *IEEE Access* **2019**, *7*, 34938–34945. [CrossRef]

16. Ansarullah, S.I.; Mohsin Saif, S.; Abdul Basit Andrabi, S.; Kumhar, S.H.; Kirmani, M.M.; Kumar, D.P. An intelligent and reliable hyperparameter optimization machine learning model for early heart disease assessment using imperative risk attributes. *J. Healthc. Eng.* **2022**, *2022*, 9882288. [CrossRef]
17. Cooney, C.; Korik, A.; Folli, R.; Coyle, D. Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech eeg. *Sensors* **2020**, *20*, 1–22. [CrossRef] [PubMed]
18. Du, X.; Xu, H.; Zhu, F. Understanding the effect of hyperparameter optimization on machine learning models for structure design problems. *CAD Comput. Aided Des.* **2021**, *135*, 103013. [CrossRef]
19. Chollet, F.; Allaire, J.J. *Deep Learning mit R und Keras: Das Praxis-Handbuch von den Entwicklern von Keras und Rstudio*; MITP-Verlags GmbH Co. KG: Frechen, Germany, 2018.
20. Elgendy, M. *Deep Learning for Vision Systems*; Simon and Schuster: New York, NY, USA, 2020.
21. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 1–48. [CrossRef]
22. Younis, M.C.; Keedwell, E. Semantic segmentation on small datasets of satellite images using convolutional neural networks. *J. Appl. Remote Sens.* **2019**, *13*, 046510. [CrossRef]
23. Zeng, S.; Zhang, B.; Zhang, Y.; Gou, J. Dual sparse learning via data augmentation for robust facial image classification. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 1717–1734. [CrossRef]
24. Abayomi-Alli, O.O.; Damaševicius, R.; Maskeliunas, R.; Misra, S. Few-shot learning with a novel voronoi tessellation-based image augmentation method for facial palsy detection. *Electronics* **2021**, *10*, 978. [CrossRef]
25. Abayomi-Alli, O.O.; Damaševičius, R.; Misra, S.; Maskeliūnas, R. Cassava disease recognition from low-quality images using enhanced data augmentation model and deep learning. *Expert Syst.* **2021**, *38*, e12746. [CrossRef]
26. Abayomi-Alli, O.O.; Damaševičius, R.; Misra, S.; Maskeliūnas, R.; Abayomi-Alli, A. Malignant skin melanoma detection using image augmentation by oversampling in nonlinear lower-dimensional embedding manifold. *Turk. J. Electr. Eng. Comput. Sci.* **2021**, *29*, 2600–2614. [CrossRef]
27. Oyewola, D.O.; Dada, E.G.; Misra, S.; Damaševičius, R. A novel data augmentation convolutional neural network for detecting malaria parasite in blood smear images. *Appl. Artif. Intell.* **2022**, *36*, 1. [CrossRef]
28. Wang, Z.; Yang, J.; Jiang, H.; Fan, X. CNN training with twenty samples for crack detection via data augmentation. *Sensors* **2020**, *20*, 4849. [CrossRef] [PubMed]
29. Hutter, F.; Hoos, H.; Leyton-Brown, K. An efficient approach for assessing hyperparameter importance. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 754–762.
30. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*; Springer Nature: Berlin, Germany, 2019; p. 219.
31. Mantovani, R.G.; Rossi, A.L.D.; Alcobaça, E.; Vanschoren, J.; de Carvalho, A.C.P.L.F. A meta-learning recommender system for hyperparameter tuning. *Inf. Sci.* **2019**, *501*, 193–221. [CrossRef]
32. Neary, P. Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Cognitive Computing (ICCC), San Francisco, CA, USA, 2–7 July 2018; pp. 73–77.
33. Ottoni, A.L.; Nepomuceno, E.G.; de Oliveira, M.S.; de Oliveira, D.C. Tuning of reinforcement learning parameters applied to sop using the Scott–Knott method. *Soft Comput.* **2020**, *24*, 4441–4453. [CrossRef]
34. Schratz, P.; Muenchow, J.; Iturritxa, E.; Richter, J.; Brenning, A. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecol. Model.* **2019**, *406*, 109–120. [CrossRef]
35. Shankar, K.; Zhang, Y.; Liu, Y.; Wu, L.; Chen, C.H. Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification. *IEEE Access* **2020**, *8*, 118164–118173. [CrossRef]
36. Pawara, P.; Okafor, E.; Schomaker, L.; Wiering, M. Data augmentation for plant classification. In Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems, Antwerp, Belgium, 18–21 September 2017; Springer: Cham, Switzerland, 2017; pp. 615–626.
37. Song, C.; Xu, W.; Wang, Z.; Yu, S.; Zeng, P.; Ju, Z. Analysis of the impact of data augmentation on target recognition for UAV-based transmission line inspection. *Complexity* **2020**, *2020*, 3107450. [CrossRef]
38. Monshi, M.M.A.; Poon, J.; Chung, V.; Monshi, F.M. CovidXrayNet: Optimizing data augmentation and CNN hyperparameters for improved COVID-19 detection from CXR. *Comput. Biol. Med.* **2021**, *133*, 104375. [CrossRef]
39. Damaševičius, R. Optimization of SVM parameters for recognition of regulatory DNA sequences. *TOP* **2010**, *18*, 339–353. [CrossRef]
40. Kalliola, J.; Kapočiūtė-Dzikienė, J.; Damaševičius, R. Neural network hyperparameter optimization for prediction of real estate prices in helsinki. *PeerJ Comput. Sci.* **2021**, *7*, e444. [CrossRef] [PubMed]
41. Połap, D.; Woźniak, M.; Hołubowski, W.; Damaševičius, R. A heuristic approach to the hyperparameters in training spiking neural networks using spike-timing-dependent plasticity. *Neural Comput. Appl.* **2022**, *34*, 13187–13200. [CrossRef]
42. Lawal, M.O. *Tomato Detection Based on Modified YOLOv3 Framework*; Springer Science and Business Media LLC: Berlin, Germany, 2021. [CrossRef]
43. Zhang, K.; Robinson, N.; Lee, S.-W.; Guan, C. *Adaptive Transfer Learning for EEG Motor Imagery Classification with Deep Convolutional Neural Network*; Elsevier BV: Amsterdam, The Netherlands, 2021. [CrossRef]

44. Roy, A.M. *Adaptive Transfer Learning-Based Multiscale Feature Fused Deep Convolutional Neural Network for EEG MI Multiclassification in Brain–Computer Interface*; Elsevier BV: Amsterdam, The Netherlands, 2022. [CrossRef]
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
46. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
47. Andriluka, M.; Pishchulin, L.; Gehler, P.; Schiele, B. 2d human pose estimation: New benchmark and state-of-the-art analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3686–3693.
48. Flusser, J.; Suk, T. Pattern recognition by affine moment invariants. *Pattern Recognit.* **1993**, *26*, 167–174. [CrossRef]
49. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90.
50. Ogundokun, R.O.; Misra, S.; Douglas, M.; Damaševičius, R.; Maskeliūnas, R. Medical Internet-of-Things Based Breast Cancer Diagnosis Using Hyperparameter-Optimized Neural Networks. *Future Internet* **2022**, *14*, 153. [CrossRef]
51. Montgomery, D.C. *Design and Analysis of Experiments*; John Wiley Sons: Hoboken, NJ, USA, 2017.
52. Luvizon, D.; Picard, D.; Tabia, H. Multi-task deep learning for real-time 3D human pose estimation and action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 2752–2764. [CrossRef]
53. Munea, T.L.; Yang, C.; Huang, C.; Elhassan, M.A.; Zhen, Q. SimpleCut: A simple and strong 2D model for multi-person pose estimation. *Comput. Vis. Image Underst.* **2022**, *222*, 103509. [CrossRef]
54. Qin, X.; Guo, H.; He, C.; Zhang, X. Lightweight human pose estimation: CVC-net. *Multimed. Tools Appl.* **2022**, *81*, 17615–17637. [CrossRef]
55. Wang, R.; Geng, F.; Wang, X. MTPose: Human pose estimation with high-resolution multi-scale transformers. *Neural Process. Lett.* **2022**, 1–24. [CrossRef]
56. Wang, W.; Zhang, K.; Ren, H.; Wei, D.; Gao, Y.; Liu, J. UULPN: An ultra-lightweight network for human pose estimation based on unbiased data processing. *Neurocomputing* **2022**, *480*, 220–233. [CrossRef]
57. Wu, Y.; Ma, S.; Zhang, D.; Huang, W.; Chen, Y. An improved mixture density network for 3D human pose estimation with ordinal ranking. *Sensors* **2022**, *22*, 4987. [CrossRef]
58. Yang, L.; Qin, Y.; Zhang, X. Lightweight densely connected residual network for human pose estimation. *J. Real-Time Image Process.* **2021**, *18*, 825–837. [CrossRef]
59. Zhang, W.; Fang, J.; Wang, X.; Liu, W. EfficientPose: Efficient human pose estimation with neural architecture search. *Comput. Vis. Media* **2021**, *7*, 335–347. [CrossRef]