Preprints.org

Article

Not peer-reviewed version

Visual Static Hand Gesture Recognition Using CNN

Ahmed Eid and Friedhelm Schwenker *

Posted Date: 9 June 2023

doi: 10.20944/preprints202306.0662.v1

Keywords: Static gesture recognition; CNN; Color model transform; Skin color segmentation; Preprocessing; Data augmentation; Adam optimizer; Cross-Entropy loss

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

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

Article

Visual Static Hand Gesture Recognition Using CNN

Ahmed Eid [1,2,*] and Friedhelm Schwenker [1,*]

[1] Institute of Neural Information Processing, Ulm University, 89081 Ulm, Germany
[2] German University In Cairo, Cairo, Egypt
* Correspondence: ahmed.sobeih@student.guc.edu.eg; Tel.: +201277877508; friedhelm.schwenker@uni-ulm.de; Tel.: +497315024159

Abstract: Hand gestures are an essential part of human-to-human communication and interaction, and therefore for technical applications, the aim is increasingly to achieve interaction between humans and computers that is as natural as possible, for example by means of natural language or hand gestures. In the context of human-machine interaction research, these methods are consequently being explored more and more. However, the realization of natural communication between humans and computers is a major challenge. In the field of hand gesture recognition, research approaches are being pursued that use additional hardware such as special gloves to classify gestures with high accuracy. Recently, deep learning techniques using artificial neural networks have been increasingly proposed for the problem of gesture recognition without using such tools. In this context, in our approach convolutional neural network (*CNN*) will be explored in detail for the task of hand gesture recognition. *CNN* is a deep neural network that can be used in the field of visual object processing and classification. The goal of this work is to recognize ten types of static hand gestures in front of complex backgrounds and different hand sizes based on raw images without the use of extra hardware. We achieved good results with a *CNN* network architecture consisting of seven layers. Through data augmentation and skin segmentation, a significant increase of the model accuracy was achieved. On public benchmarks the ten gestures have been classified almost perfectly with a testing accuracy of 96.5%

Keywords: static gesture recognition; *CNN*; color model transform; Skin color segmentation; preprocessing; data augmentation; adam optimizer; cross-entropy loss

1. Introduction

In order to communicate with computers in more effective and natural ways, people have experimented with various methods for more than half of the last century. Over time, human interaction mainly was made via a keyboard and mouse. Most of the time we interact with computers, we use our fingers and eyes, but other body parts, including our legs, arms, and mouth are underutilized or never used at all. This is frustrating since it's like composing emails with just one finger. Standard image processing methods do not produce excellent results, so machine learning is required for gesture detection to reach its full potential. Gestures are meaningful, expressive body motions involving physical movements of the body, hands, arms, head, face, or fingers. Gesture recognition is the process that seeks to identify gestures and translate them into commands that can facilitate effective communication between humans and computers. Hand gestures are either dynamic or static. Implicit hand gestures and postural behavior are considered for the recognition of emotional states [1,2], but this is not the topic of this paper. Here, we focus on the recognition of static explicit hand gestures.

At present, there are some problems in visual gesture recognition, such as accuracy, real-time or poor robustness. Although there are many methods of gesture recognition, vision-based gesture recognition still faces many serious problems in practice. It is mainly reflected in low recognition rate, poor robustness, insensitivity in real-time and poor practicability, Gesture recognition should bring great results no matter the background, it should work whether you're in the car, at home, or

© 2023 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

walking down the street. Using convolutional neural networks *CNN* helps to overcome the problem of identifying the gesture in complex backgrounds with very high accuracy.

Advancements in convolutional neural networks (*CNN*) and the recently emerged deep learning techniques avoids the need of deriving complex hand crafted feature descriptors from images, so it outweighs the classical approach to hand gesture recognition [3–6]. By learning the high level abstractions in images, CNNs automate the feature extraction process and use hierarchical architecture to capture the most discriminative feature values [7,8].

Pisharady et al.[9] have suggested a skin-based approach to identify hand regions in pictures. Additionally, an *SVM* classifier was used to recognize the hand motions. In [10] Gao et al. proposed a parallel *CNN* model which used *RGB* and depth images as input. Parallel *CNN* consists of two *CNN*s, namely depth-*CNN* and *RGB*-*CNN*, where one takes a depth image as input while the other takes an *RGB* image as input. A prediction probability weighted the last layer of each *CNN* output and concatenated to be the input to a softmax classifier layer, the model accuracy has reached 93.3% for american sign language.

In [11] Oliveira et al. proposed *CNN* with four convolutional layers, with a max-pooling layer affixed to each convolutional layer, followed by a fully connected layer and softmax classifier. The proposed *CNN* was able to attain 99% for Irish Sign Language (ISL). The very high accuracy can be explained by the simple image black backgrounds which makes it so easy and unchallenging to identify images and gestures. In contrast, this is not the case in the real world where images have background noises and changes in illumination. Arenas et al. [12] derived a *CNN* architecture from Directed A cyclic Graph (*DAG*) structure (*DAG*-*CNN*). A self-constructed dataset was used to experiment the model on, the dataset consists of 10 gestures for controlling the robotic arm, and model accuracy was 84.5%.

Using fully connected layers of a pre-trained artificial neural network (AlexNet), Sahoo et al. [13] proposed a deep *CNN* feature-based static hand gestures recognition system. This system reduces redundant features using principal component analysis after deep features are extracted using fully connected layers of AlexNet (*PCA*). A support vector machine (*SVM*) was then used as a classifier to categorize the poses of hand motions. The American Sign Language (*ASL*) dataset was used to test the system's performance on 36 gesture postures, and the average accuracy score was 87.83%. In [14] Wadhawan et al. proposed a generic *CNN* architecture for static sign language recognition. The network had an accuracy of 98.85% on the Indian sign language dataset. The dataset consists of images captured in a white simple background that can be identified as relatively unchallenging. The network consists of two convolutional layers, a max pooling layer including the dropout, and two fully connected layers.

For the purpose of assessing human behavior in the context of classroom learning and instruction with two teachers, Wang et al. [15] introduced a recognition model of hand gestures based on CNN. The analysis of the teacher's nonverbal behaviors that improve the learning results of learners and attract their attention is done by exploiting the recognized instructors' hand gestures. A non-linear neural network with four convolutional layers is used in this model to extract the features of photos of hand gestures. For achieving robust recognition, three convolution layers of *CNN* are designed. A dataset of 38425 infrared hand gesture images is used to test and evaluate the model. These images which represent the extracted key frames from the infrared videos have two labels, pointing and non-pointing gestures. A split of 80% for the training and 20 % for the test has been done on the infrared hand gestures. The model has reached an obtained ratio of recognition accuracy of more than 92%.

In [16], Zuocai Wang et al. suggested a method for identifying hand gestures through the use of particle filtering. By implementing this filtering technique on hand gesture images with identical backgrounds, the researchers achieved an accuracy of 90.6 %. In [17], Suguna and Neethu utilized shape features obtained from hand gesture images to categorize them into different classes. The extracted features were then taught and sorted into clusters using the k-means clustering algorithm.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

In [18], Marium et.al put forward a method for hand gesture recognition, which involved the use of a convexity algorithm approach. The researchers tested this technique on hand gesture images with identical backgrounds and obtained an accuracy of 87.5 %. However, the approach was limited by the fact that the algorithm worked best when the background of the hand gesture image was stationary.

In [19], Ashfaq and Khurshid converted spatial domain format hand gesture images into multi-class domain format images using the Gabor filtering approach. They then employed both Bayesian and Naive Bayes classifiers to categorize the test hand gesture images into different classes. The researchers found that the Naive Bayes classifier produced higher levels of classification accuracy than the Bayesian classification methodology, owing to its straightforward architecture pattern. In [20], Rahman and Afrin used a support vector machine (SVM) classification approach in 2013 to categorize hand gesture images into various classes. The researchers achieved a sensitivity of 89.6%, a specificity of 79.9%, and an accuracy of 85.7%. However, the error rate was high in this method, and it was not suitable for fast-moving background and foreground object images.

In [21,22], Authors employed utilized Naive Bayes classifier and support vector machine (SVM) approaches for recognizing gestures, but these methods were unable to handle large training datasets and needed a high number of training samples. To overcome these limitations, the current study introduces a CNN classifier, which does not require a high number of training samples and has a low complexity level. In [23], Rao et al. developed a hand gesture recognition system using a hidden Markov model. The authors constructed a Markov model for foreground fingers in a hand gesture image. This Markov model was used in both the training and testing modes of the binary classification approach. The authors produced 90.1%.

In [24], the hand postures are classified using the shape, texture, and color features, with a support vector machines classifier. The proposed system utilizes a Bayesian model of visual attention to generate a saliency map and to detect and identify the hand region and they reported an accuracy of 94.36 % on the NUS II dataset that we use in our research. In [25], the authors proposed a *CNN* model with two convolutional layers, two max-pooling layers, and one last fully connected layer. Dropout and activation functions are optional, However, they reported better results using both of them. The best accuracy reported on the NUS-II dataset was 89.1% by adding the dropout and activation functions. In our research paper we will compare our results with the ones in [24,25] as they reported results on the same challenging dataset we use and they also use *CNN*.

In this research, we propose a convolutional neural network *CNN* to recognize static hand gestures in complex backgrounds. The objective is to increase the accuracy of correctly identified gestures. Testing the accuracy of the model is done by comparing the true label of the image and the predicted label.

The efficiency of deep learning in extracting and classifying high-level aspects of data has recently been the focus of existing research.

In summary contributions of this paper are adding the power of newly proposed preprocessing techniques in this research for the images using skin segmentation and data augmentation with the power of using CNN for classifying images. To the limit of our knowledge, there has been no previous publication that has performed skin segmentation on the NUS II dataset using the same methodology we performed. Also, there is no paper that reported any results of combining both using *CNN* model and using skin segmentation on a complex dataset as NUS II dataset as we performed.

We compare our accuracy to the one in [25] as it uses *CNN* and the same NUS II dataset that we use. The accuracy of the proposed *CNN* model has improved from the one in [25] going from 89.1 % to 96.5 % which means the loss has decreased by 67.9 %.

In section 2, the concepts of skin segmentation data augmentation, cross-entropy loss function, and the structure of the newly proposed CNN are introduced. We also discuss the training details and analysis of different methods and tools used in different training experiments. In section 3, we discuss the results of the proposed experiments. Finally, in Section 5 we discuss how the results can be interpreted from the previous study in [25] as it has used the same dataset and a *CNN*.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

2. Materials and Methods

2.1. Introduction

In spite of the advances in image processing techniques and gesture recognition, an essential challenge is still unsolved: how can we recognize gestures in complex backgrounds without using hardware with very high accuracy? It turns out that using a machine learning model called *CNN* helps us a lot to achieve this by using the right dimensions for the *CNN* layers and choosing the best optimization techniques, we can achieve very high results.
Identifying the right label for the test image is the optimization aim of our *CNN* model. we can achieve this by minimizing the loss function for the training and validation datasets to reach as much accuracy as possible. To train the model Skin segmentation is used to reduce the data in the image and remove unwanted pixels that do not have skin *HSV* values. The challenge with this technique is that some pixels in the background have skin colors already but it helps in reducing the data of the image and reduces the loss significantly.

Softmax is used to measure the predicted probabilities of each class by giving them values between 0 and 1. As a consequence of using softmax, to measure the loss, the Cross-Entropy loss function is used to measure the loss for the training dataset. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. Data augmentation is used to increase the dimensions of our training dataset but validation and test datasets are not affected and it has shown a significant increase in the accuracy and decrease in the loss function as it helps in reducing overfitting and giving the model more data to train on.

2.2. Structure of the CNN

As shown in Figure 1, the *CNN* model consists of 2 convolutional layers followed by *Relu* activation function for each one, and 2 pooling layers apart from the input and the output layer. As shown in Figure 1, the input image of 32×32 pixels is convolved with 15 filter maps of size 6×6 to produce 15 output maps of 29×29 in layer 1. These output maps are operated upon with a *Relu* activation function. Then downsampling of the output convolutional maps is done with max-pooling of 2×2 regions to yield 15 output maps of 14×14 in layer 2. The 10 output maps of layer 2 are convolved with each of the 30 kernels of size 3×3 to obtain 30 maps of size 14×14. These maps are further downsampled by a factor of 2 by max-pooling to produce 30 output maps of size 7×7 of layer 4. The output maps from layer 4 are concatenated to form a single vector during training and fed to the next layer which is the fully connected layer. A dropout probability of 0.5 is used to reduce overfitting.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

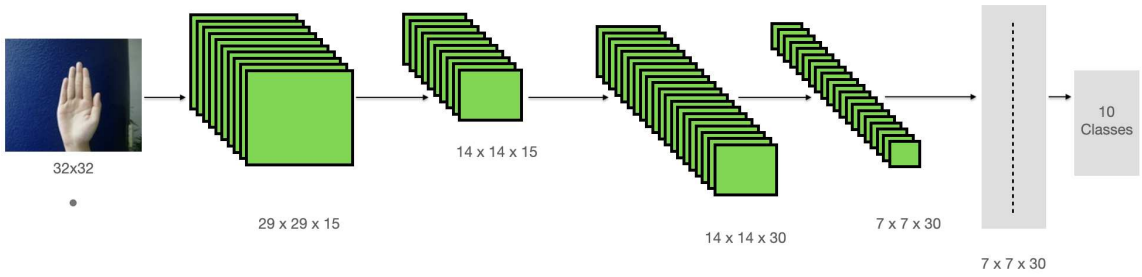

Figure 1. Layers of the proposed *CNN* model that shows each layer with its dimensions, the input size, and the number of output classes. The first layer is a convolutional layer with 15 output maps with dimensions of 29×29, The second one is a max-pooling layer which has dimensions of 14×14, The third layer is a convolutional layer with 30 output maps with dimensions of 14×14, The fourth layer is a max-pooling layer with dimensions of 7×7, The fifth layer is a fully connected layer which is a single vector of all the output maps in the previous layer.

2.3. Skin segmentation

The first technique we use to try to make the model learn better and increase its accuracy is skin segmentation which will be used in the second experiment. Segmentation aims at partitioning areas in the image based on color, shape, or textures. It is useful in many computer vision applications such as medical image analysis, object detection and recognition, and content-based image retrieval *CBIR* [26].

The skin segmentation technique is done using the OpenCV Python Binary extension loader module. The segmentation is done by converting the image first from *RGB* to *HSV* color space to be able to extract the hue values for skin colors. In the OpenCV library in Python, the measuring unit of hue values in the *HSV* (Hue, Saturation, Value) color space is typically 8-bit unsigned integers, ranging from 0 to 179. This means that hue values are quantized into discrete values, where each integer represents a specific hue bin or hue category. The 0 to 179 range is used to represent the full 360 degrees of hue values typically used in other systems, where each bin or category corresponds to a hue angle increment of approximately 2 degrees (360 degrees divided by 180 bins). This integer representation allows for efficient storage and processing of hue values in digital images and computer vision applications.

Hue values for skin colors are found to be ranging from 0 to 38 which means they are ranging from 0 to 76 degrees in the normal HSV color space. The saturation and value(Also known as brightness) are given a complete range from 0 to 255. This is done to overcome the complex background which has a wide range of brightness values that can be very dark or very bright. We only truncate the values for hue which give us the true skin pixels whatever the value of the saturation of light or brightness. Note also that images have a complex background which includes skin-colored surfaces. So, any skin-colored surface in the background is included in the image. But overall it is better than the normal image as there is a lot of unwanted pixels in the background that are removed which makes us use only pixels that we can learn the most from during the training and validation phase.

2.4. Data augmentation

Data augmentation is a very powerful technique for constructing better datasets. Many augmentations have been proposed which can generally be classified as oversampling techniques. Deep learning models rely on big data generated from data augmentation to avoid overfitting. Artificially inflating datasets using the methods of data augmentation achieves the benefit of big data in the limited data domain. [27]

Data augmentation is used on our dataset to increase its size by 10 times. The training data set was 1602 images before augmentation and then after being augmented it becomes 16020 which is 10

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

times the size of the original one which helps in reducing overfitting by giving more data to train on. The data augmentation is performed by randomly rotating each image by an angle that ranges from 20 degrees to the left to 20 degrees to the right.

2.5. Dropout

Dropout is a regularization technique that helps to address the issue of overfitting in neural networks. During training, standard backpropagation learning can result in brittle co-adaptations that only work well on the training data and do not generalize to new, unseen data. By randomly dropping out or deactivating a fraction of hidden units during each training iteration, dropout disrupts these co-adaptations, making the presence of any particular hidden unit less reliable. This technique has been found to be effective in improving the performance of neural networks in various application domains, such as object classification, digit recognition, speech recognition, document classification, and computational biology data analysis [28]. In our proposed *CNN*, The dropout is added at the end of our model before the last fully connected layer to reduce the overfitting that is reaching the final layer and make the model generalizes to new, unseen data.

2.6. Loss Function

Also called logarithmic loss, log loss, or logistic loss. Each predicted class probability is compared to the actual class desired output of 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value. The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0.

Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss, i.e, the smaller the loss the better the model. A perfect model has a cross-entropy loss of 0. Cross Entropy is defined as follows:

$$L = -\sum_{1}^{n} p_i * log(q_i) \tag{1}$$

here n is the number of classes, p_i the true probability of i_{t_h} class in target, and q_i the softmax probability for the i_{t_h} class.

Here if we look at equation 1, we can see that loss is calculated for each training example by calculating the negative natural logarithm for the output of the softmax function (q_i) multiplied by the true probability of the target which is 0 or 1 (p_i). After calculating the loss for every training example, we sum all of them to get the training loss of the whole training dataset.

2.7. Training Details

All the experiments were done on Google colab CPU. Before training the images are resized to 32x32. A batch size of 64 is used for the training set and a batch size of 32 is used for both the validation set and the test set. Each image is resized into 32x32 and then enters the first convolution layer for the training phase. A learning rate of 0.001 is used with *Adam* optimizer as our optimization function. The original dataset is separated into 2 phases. In the first phase, it is shuffled randomly and then separated in train and test sets with maintaining the ratio of each class in both of them as the same ratio of splitting by using stratify in the train_test_Split function used to split them. In the second phase, the training set from the first phase is split again into training and validation sets by also maintaining the same ratio of classes in both of them as we did in the first phase by using Stratify.

NUS II Dataset is the dataset used for the training. This is a 10-class hand posture dataset as shown in Figure 2. The postures are shot in and around the National University of Singapore (NUS), against complex natural backgrounds, with various hand shapes and sizes which makes it a challenging dataset as shown in Figure 3. The postures are performed by 40 subjects, of different ethnicities, against different complex backgrounds. The subjects include both males and females in the

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

age range of 22 to 56 years. The subjects are asked to show the 10 hand postures, 5 times each. They are asked to loosen the hand muscle after each shot, in order to incorporate the natural variations in the postures. The dataset consists of 2000 images. Each image has a size of 160 x 120. The dataset is developed for testing the proposed algorithm in [24].

Figure 2. Sample images from NUS hand posture dataset-II, showing posture classes 1 to 10 [24]

Figure 3. Sample images (class 9) from NUS hand posture dataset-II, showing the variations in hand posture sizes and appearances [24]

2.8. Training experiments details

These are the details of the experiments that have been conducted and will be referenced in the following sections. All the experiments have been done on *NUS II* dataset and the model has been trained for 200 epochs using *Adam* optimizer. In all experiments except for the last experiment which uses data augmentation, The training dataset is composed of 1602 images divided into batches of size 64 for each batch making a total number of 26 batches where each of the first 25 batches contains 64 images and the last batch contains only 2 images. For the experimnent that uses data augmentation, The training dataset consists of 16020 images divided into batches of size 64 for each batch making a total number of 251 batches where each of the first 250 batches contains 64 images and the last batch contains only 20 images. In all the experiments including the one where we use data augmentation, the validation dataset consists of 198 images that are divided into batches of size 32 for each batch making a total number of 7 batches where each of the first 6 batches contains 32 images and the last batch contains 6 images only. The test dataset consists of 200 images that represent exactly 10 % of the total size of our dataset. The test dataset is divided into batches of size 32 making a total number of 7 batches where each of the first 6 batches contains 32 images and the last batch contains only 8 images. As we can see we did the data augmentation only on the training dataset because we need the validation and test datasets not touched to be able to validate and test on the original data that will mirror the data will be seen in real life. In the train dataset we keep the same portion of classes using Stratify parameter in the train_test_split method in scikit learn model_selection library.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

2.9. Cross-validation Details

In the validation dataset we also keep the same portion of classes using stratify as we have seen in the training dataset. Validation is done by evaluating our model on the validation dataset using the eval method in model library in ptyhon. The prediction of the validation is done on every image by iterating over them and computing the gradient by giving both the prediction and the original labels to a criterion function which computes the gradient from the cross-entropy loss function that we used. Then after getting the validation loss we do backpropagation to learn how to update the weights and get a better validation loss for the next iteration. A softmax function is then applied to the predicted variables to make the predicted variables for each class sum up to 1. The cross-validation accuracy is done by enumerating over the validation data loader and then comparing the original labels of the image with the labels predicted from our model and then after every iteration, we check if the original label equals the predicted label if yes we increment our variable and then see at the end how much is our testing accuracy by dividing the number of positive predictions by the number of images of the validation dataset as follows:

$$Validation\ accuracy = \frac{number\ of\ positive\ prediction}{number\ of\ images\ in\ the\ validation\ dataset}$$

After every epoch we see if the validation loss is less than the least validation loss so far we save a new model state dictionary and we update our variable to be the least validation loss for the next epoch to do the same thing again so when we finish we can know which is the best weights to use for testing phase.

2.10. Testing Details

The testing dataset has been derived from the original NUS II dataset by shuffling the dataset and taking 10 % of it keeping the same portions of each class in the testing set to be able to do equal testing for all the classes also to be able to train all the classes by the same number of images so the best option for both the training and testing phase for the most accurate results. Before the testing we load the state dictionary of the model in which we have saved the best parameters to use while the training and validation phases and then we use the eval method to evaluate our model first. A softmax function is applied to the predicted variables to make the predicted variables for each class sum up to 1. The testing is done by enumerating over the test data loader and then comparing the original labels of the image with the labels predicted from our model and then after every iteration we check if the original label equals the predicted label if yes we increment our variable and then see at the end how much is our testing accuracy by dividing the number of positive predictions by the whole number of images in the test dataset as follows :

$$Test\ accuracy = \frac{number\ of\ positive\ prediction}{number\ of\ images\ in\ the\ test\ dataset}$$

3. Results

This section will go over the validation loss, training loss, and confusion matrix of each experiment and will show how the techniques used will affect the accuracy of the model to get the highest accuracy possible.

3.1. Results of using only the structure of CNN without skin segmentation or data augmentation or dropout

The accuracy of our first experiment is 82.5 %. The reason for this low accuracy is that the model overfits when we do not use dropout. As we can see in Figure 4, the overfitting can be seen in the big difference between training loss and validation loss. The reason behind that overfitting is that our model is performing so well on the training data but does not perform well on the new data. This overfitting occurs when the model is using too many features and not enough amount of data. To solve the two problems we use later the dropout rate to reduce the features of the model and we

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

will solve the problem of increasing the size of the dataset later when we use the data augmentation. Also the accuracy here is not so high, we did this experiment on purpose before using any additional techniques or dropout to show how necessary it is to add them later.

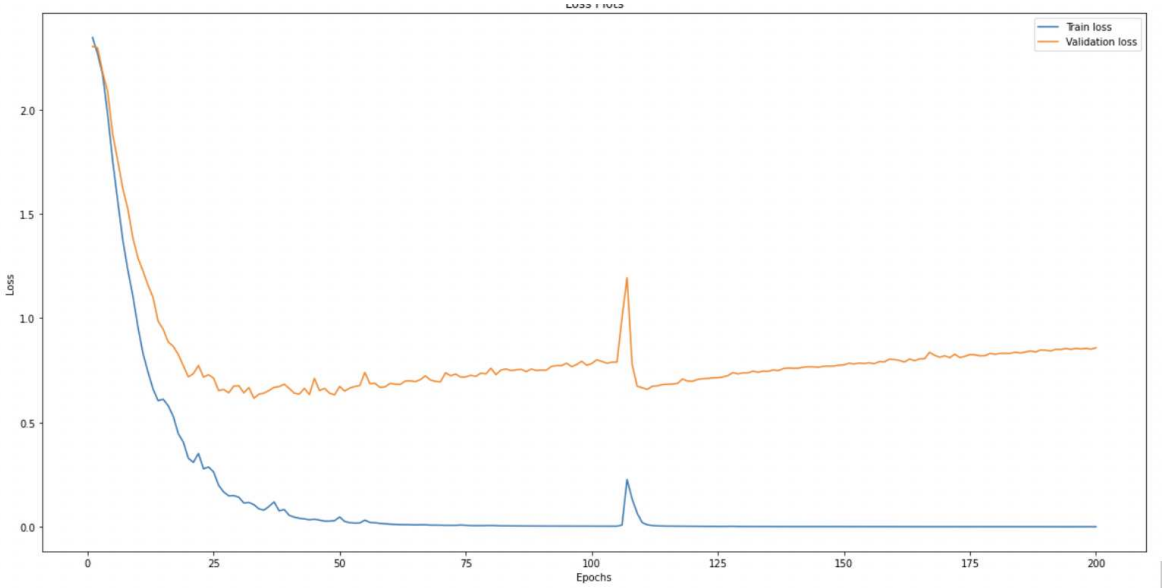

Figure 4. The figure shows the training and validation loss of the *CNN* model without using dropout.

Now to see how the model performs in each class we can look at the confusion matrix and see how many images the model performed well on and how many images the model misclassified. we can also observe a lot of misclassified photos due to overfitting. As we can see in Table 1, the model performs very badly when classifying the class (H) and class (I), every class consists of 20 photos. The model performs the best on class (J) as it misclassifies only one photo to predict it as (I) instead of (J) but all other 19 photos of the (J) class have been classified correctly.

Table 1. This table shows confusion matrix of not using dropout or skin segmentation or data augmentation with an accuracy of 82.5 %

Confusion matrix with an accuracy of 82.5 %										
	A	B	C	D	E	F	G	H	I	J
A	17	0	0	2	0	0	0	1	0	0
B	1	18	0	0	1	0	0	0	0	0
C	0	0	16	2	0	0	0	1	1	0
D	2	0	0	17	0	0	1	0	0	0
E	0	0	0	0	18	0	1	0	1	0
F	0	0	0	0	1	18	0	1	0	0
G	0	0	1	1	0	0	18	0	0	0
H	2	1	1	3	1	1	0	10	1	0
I	0	0	0	1	1	3	0	0	14	1
J	0	0	0	0	0	0	0	0	1	19

3.2. Results of adding dropout to the CNN model

The accuracy of adding dropout to the *CNN* model is 90.5 %. As we can see in Figure 5, by using a dropout rate of 0.5, the difference between the training loss and the validation loss has decreased significantly which means that overfitting has decreased as the model now generalizes better on the new data used for validation or test. Also we can see that the overall accuracy of the model increased from 82.5 % to 90.5 %. As we can see in Table 2, the model improves so much with most of the classes, especially with class (H) which seems to be overfitting more than other classes as it has increased from

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

10 to 14 which means it has classified 4 images more correctly. The classes (B, E, G) have been classified correctly for all of the images after doing the dropout as they have an accuracy of 100 %. So we can see that using dropout has improved the model so much.

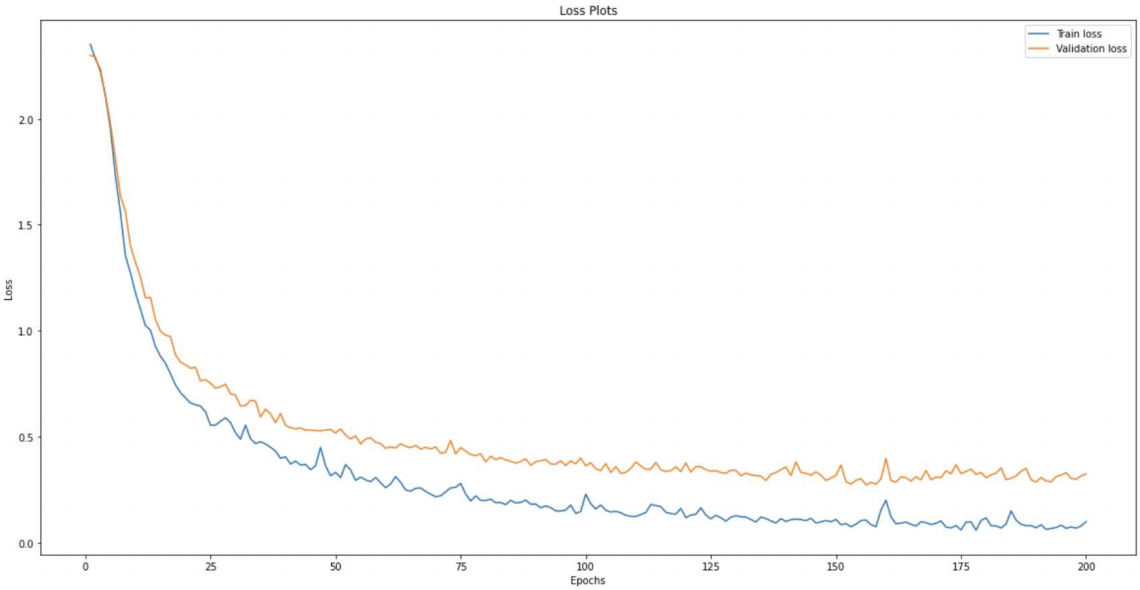

Figure 5. The figure shows the training and validation loss of the *CNN* model after adding dropout.

Table 2. This table shows the confusion matrix of using dropout and not using skin segmentation or data augmentation

Confusion matrix with an accuracy of 90.5 %										
	A	B	C	D	E	F	G	H	I	J
A	17	0	1	0	0	0	1	1	0	0
B	0	20	0	0	0	0	0	0	0	0
C	0	1	19	0	0	0	0	0	0	0
D	0	1	0	19	0	0	0	0	0	0
E	0	0	0	0	20	0	0	0	0	0
F	0	0	2	0	1	17	0	0	0	0
G	0	0	0	0	0	0	20	0	0	0
H	1	1	1	1	0	0	0	14	2	0
I	0	0	0	0	1	3	0	0	16	0
J	0	0	0	0	0	0	0	0	1	19

3.3. Results of using skin segmentation

As a result of using skin segmentation, alot of unwanted pixels have been removed from each image making it easier for the *CNN* model to correctly classify images as we will see later in the confusion matrix of this experiment in Table 3. In Figure 6 we can see an image example form our dataset before and after segmentation. we can see in the figure that the pixels that have the skin color are all present in the segmented image along with some other pixels that has hue values close to the skin color but we can see clearly that a lot of pixels in the upper part of the image have been removed.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

Table 3. This table shows the confusion matrix of using skin segmentation and not using data augmentation

Confusion matrix with an accuracy of 93.5 %										
	A	B	C	D	E	F	G	H	I	J
A	19	0	1	0	0	0	0	0	0	0
B	0	18	1	0	0	0	0	1	0	0
C	0	0	19	0	1	0	0	0	0	0
D	1	0	0	19	0	0	0	0	0	0
E	0	0	0	0	18	1	0	0	1	0
F	0	0	0	0	0	20	0	0	0	0
G	0	0	0	0	1	0	19	0	0	0
H	0	2	0	0	1	0	0	16	1	0
I	0	0	0	0	1	0	0	0	19	0
J	0	0	0	0	0	0	0	0	0	20

Figure 6. The figure shows an original image and the segmented image

As we can see in Table 3, using skin segmentation has improved the accuracy of our model and decreased the loss of training and validation loss. The accuracy has improved from 90.5 % without using skin segmentation to 93.5 % by using skin segmentation which means that our model has improved by nearly 31.6 %. As the test dataset consists of 200 images so, each image represents 0.5 %. As a result, saying that the model accuracy increases from 90.5 % to 93.5 % means that 6 more images in the test dataset have been classified correctly. The reason behind that is that much-unwanted data has been removed from every image leaving only the pixels that have the skin color values. Due to the complex and challenging background, We can see that some pixels in the background have skin color too which means that they are included in our image with hands. With a simpler dataset that has a clear background, the improvement in the accuracy is much more as the image simply consists of only hands. But we can say also with these complex backgrounds an improvement of 31.6 % is pretty high and shows that skin segmentation has a very high effect. Now let us take a look at the confusion matrix and see which images have been classified correctly.

As shown in Table 3, many classes have improved and more images are classified correctly than in the first experiment. The image of class (F) has increased from 17 to 20 which means all images of class (F) are predicted correctly after doing skin segmentation. Also class (J) now has an accuracy of 100 % as all its images are predicted correctly. Classes with the lowest accuracy from the first experiment (I, H) have increased their accuracy significantly as we can see class (H) has increased from 14 to 16 which means that two more images are classified correctly. Looking at class (I) we can observe that it has increased from 16 to reach 19 which means that 3 more images are classified correctly.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

3.4. Results of using skin segmentation and data augmentation

As a result of using data augmentation, each segmented image in the training dataset is augmented to 10 different images using rotation by an angle that ranges from 20 degrees to the left to 20 degrees to the right. Here in Figure 7, we can see an example of two original images and the 10 augmented images for each image

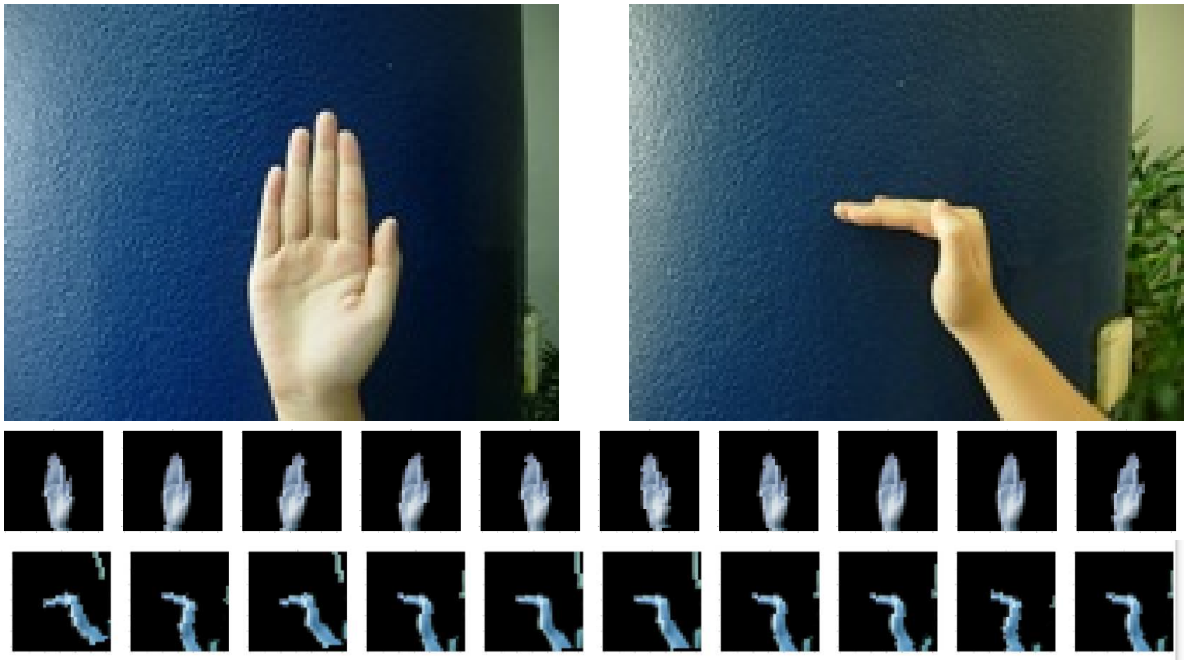

Figure 7. The figure shows an example of data augmentation implemented with two original images and their corresponding augmented images (10 augmented images for each original image).

As we can see in Table 4, using data augmentation has improved the accuracy of our model and decreased the loss of training and validation loss. The accuracy has been enhanced from 93.5% without using skin segmentation to 96.5% by using data augmentation which means that our model has improved by nearly 41.6%. Increasing the accuracy by 3% means that 6 more images have been classified correctly by using data augmentation. The reason behind that is the increasing of the size of our training dataset to 10× of its original size and decreasing the overfitting.

Table 4. This table shows the confusion matrix of using skin segmentation and data augmentation

Confusion matrix with an accuracy of 96.5 %										
	A	B	C	D	E	F	G	H	I	J
A	19	0	1	0	0	0	0	0	0	0
B	0	20	0	0	0	0	0	0	0	0
C	0	0	20	0	0	0	0	0	0	0
D	0	0	0	20	0	0	0	0	0	0
E	0	0	0	0	19	1	0	0	0	0
F	0	0	0	0	0	19	0	0	0	1
G	0	0	0	0	0	0	20	0	0	0
H	1	0	0	1	1	0	0	17	0	0
I	0	0	0	0	1	0	0	0	19	0
J	0	0	0	0	0	0	0	0	0	20

As shown in Table 4, we can see that the model has improved significantly as shown some classes (B, C, D, G, J) have improved in a way that all their images are classified correctly using both data

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

augmentation and skin segmentation. Other classes have only one image misclassified and all other 19 images are classified correctly as we can see in classes (A, E, F, I). We can see here that the class with the least accuracy in all the experiments which is class (H) has one more image classified correctly to have a total of 17 correctly classified images instead of 16 images in the previous experiment.

4. Conclusion

As we can see in Table 5, After conducting 4 training experiments, the model has proven to perform the best by using the our the proposed *CNN* model parameters with adding skin segmentation and data augmentation which has decreased the overfitting to reach an accuracy of 96.5% at the end. Some classes are more difficult to classify than other ones which make some classes predict correctly for all their images in the test dataset but other ones have some images misclassified which depends on the class that the hand gesture itself is hard to recognize. This can return to the more complex the details of the hand gesture is and also that some images are more clear than others.

But overall reaching 96.5% accuracy is pretty good on this dataset with complex backgrounds.

Table 5. This table shows the summary of our experiments and their accuracies to do the conclusion

Experiments and their accuracies	
Experiment	Accuracy
Using only the structure of CNN Without skin segmentation or data augmentation or dropout	82.5%
Adding dropout	90.5%
Adding skin segmentation and dropout	93.5%
Adding data augmentation, skin segmentation and dropout	96.5%

5. Discussion

We will discuss our results and compare them to the methods reported on the same dataset in [24,25]. As we can see in Table 6, our proposed model shows an improvement in the accuracy as it reported higher accuracy than the state-of-the-art methods that tried to recognize hand gestures of the same dataset. we will compare more to one in [25], as it uses also *CNN* model. The proposed neural network parameters are modified from the ones introduced in [25]. As an example, the filter size in the proposed network of the first layer is bigger than the one introduced in [25], and this is done for the purpose of increasing the receptive field as The filter size determines the area of the input image that each convolutional operation considers. A larger filter size allows the convolutional layer to capture larger patterns or features in the input image. Adding the proposed skin segmentation technique in this research has increased the model accuracy significantly compared to the accuracy in [24,25] and has shown the importance of preprocessing and how it makes a big difference compared to the results.

Table 6. This table shows the comparison of the Recognition accuracy of the Proposed Method on NUS II Hand Posture Dataset

Comparison of the proposed model accuracy with the state of the art		
Author	Method	Accuracy
Pramod et al. [24]	C2SMF (color,shape and texture)/ multiclass SVM	94.36 %
Mohanty et al. [25]	Deep Learning with CNN	90.5%
proposed	Deep Learning with CNN	96.5%

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

References

1. Amirian, M.; Kächele, M.; Palm, G.; Schwenker, F. Support vector regression of sparse dictionary-based features for view-independent action unit intensity estimation. In Proceedings of the 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017). IEEE, 2017, pp. 854–859.
2. Hihn, H.; Meudt, S.; Schwenker, F. On gestures and postural behavior as a modality in ensemble methods. In Proceedings of the Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR 2016, Ulm, Germany, September 28–30, 2016, Proceedings 7. Springer, 2016, pp. 312–323.
3. Neto, G.M.R.; Junior, G.B.; de Almeida, J.D.S.; de Paiva, A.C. Sign language recognition based on 3d convolutional neural networks. In Proceedings of the Image Analysis and Recognition: 15th International Conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15. Springer, 2018, pp. 399–407.
4. Li, G.; Tang, H.; Sun, Y.; Kong, J.; Jiang, G.; Jiang, D.; Tao, B.; Xu, S.; Liu, H. Hand gesture recognition based on convolution neural network. *Cluster Computing* **2019**, *22*, 2719–2729.
5. Tao, W.; Leu, M.C.; Yin, Z. American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion. *Engineering Applications of Artificial Intelligence* **2018**, *76*, 202–213.
6. Xing, K.; Ding, Z.; Jiang, S.; Ma, X.; Yang, K.; Yang, C.; Li, X.; Jiang, F. Hand gesture recognition based on deep learning method. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE, 2018, pp. 542–546.
7. Traore, B.B.; Kamsu-Foguem, B.; Tangara, F. Deep convolution neural network for image recognition. *Ecological informatics* **2018**, *48*, 257–268.
8. Affonso, C.; Rossi, A.L.D.; Vieira, F.H.A.; de Leon Ferreira, A.C.P.; et al. Deep learning for biological image classification. *Expert systems with applications* **2017**, *85*, 114–122.
9. P. K. Pisharady, P.V.; Loh, A.P. Attention based detection and recognition of hand postures against complex backgrounds. *International Journal of Computer Vision* **2013**, *101*, 403–419.
10. Gao, Q.; Liu, J.; Ju, Z.; Li, Y.; Zhang, T.; Zhang, L. Static hand gesture recognition with parallel CNNs for space human-robot interaction. In Proceedings of the International Conference on Intelligent Robotics and Applications. Springer, 2017, pp. 462–473.
11. Oliveira, M.; Chatbri, H.; Little, S.; Ferstl, Y.; O'Connor, N.E.; Sutherland, A. Irish sign language recognition using principal component analysis and convolutional neural networks. In Proceedings of the 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA). IEEE, 2017, pp. 1–8.
12. Arenas, J.O.P.; Moreno, R.J.; Beleño, R.D.H. Convolutional neural network with a dag architecture for control of a robotic arm by means of hand gestures **2018**.
13. Sahoo, J.P.; Ari, S.; Patra, S.K. Hand gesture recognition using PCA based deep CNN reduced features and SVM classifier. In Proceedings of the 2019 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS). IEEE, 2019, pp. 221–224.
14. Wadhawan, A.; Kumar, P. Deep learning-based sign language recognition system for static signs. *Neural computing and applications* **2020**, *32*, 7957–7968.
15. Wang, J.; Liu, T.; Wang, X. Human hand gesture recognition with convolutional neural networks for K-12 double-teachers instruction mode classroom. *Infrared Physics & Technology* **2020**, *111*, 103464.
16. Wang, Z.; Chen, B.; Wu, J. Effective inertial hand gesture recognition using particle filtering based trajectory matching. *Journal of Electrical and Computer Engineering* **2018**, *2018*, 1–9.
17. Suguna, R.; Neethu, P. Hand gesture recognition using shape features. *Int. J. of Pure and Applied Mathematics* **2017**, *117*, 51–54.
18. Marium, A.; Rao, D.; Crasta, D.R.; Acharya, K.; D'Souza, R. Hand gesture recognition using webcam. *American Journal of Intelligent Systems* **2017**, *7*, 90–94.
19. Ashfaq, T.; Khurshid, K. Classification of hand gestures using Gabor filter with Bayesian and naïve Bayes classifier. *International Journal of Advanced Computer Science and Applications* **2016**, *7*.
20. Rahman, M.H.; Afrin, J.; et al. Hand gesture recognition using multiclass support vector machine. *International Journal of Computer Applications* **2013**, *74*, 39–43.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2023 doi:10.20944/preprints202306.0662.v1

21. Park, S.; Yu, S.; Kim, J.; Kim, S.; Lee, S. 3D hand tracking using Kalman filter in depth space. *EURASIP Journal on Advances in Signal Processing* **2012**, *2012*, 1–18.
22. Ren, Z.; Yuan, J.; Meng, J.; Zhang, Z. Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia* **2013**, *15*, 1110–1120.
23. Rao, J.; Gao, T.; Gong, Z.; Jiang, Z. Low cost hand gesture learning and recognition system based on hidden markov model. In Proceedings of the 2009 Second International Symposium on Information Science and Engineering. IEEE, 2009, pp. 433–438.
24. Pisharady, P.K.; Vadakkepat, P.; Loh, A.P. Attention based detection and recognition of hand postures against complex backgrounds. *International Journal of Computer Vision* **2013**, *101*, 403–419.
25. Mohanty, A.; Rambhatla, S.S.; Sahay, R.R. Deep gesture: static hand gesture recognition using CNN. In Proceedings of the Proceedings of International Conference on Computer Vision and Image Processing. Springer, 2017, pp. 449–461.
26. Dwina, N.; Arnia, F.; Munadi, K. Skin segmentation based on improved thresholding method. In Proceedings of the 2018 International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON), 2018, pp. 95–99. https://doi.org/10.1109/ECTI-NCON.2018.8378289.
27. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *Journal of big data* **2019**, *6*, 1–48.
28. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* **2014**, *15*, 1929–1958.

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