Constrained Convolutional Neural Network Models for Optimizing Fully Connected Layer Weights in CNNs

Ugur Talas
Institute of Graduate Education Engineering
Bilecik Seyh Edebali University
Turkiye
ugur.talas@bilecik.edu.tr

Burakhan Cubukcu
Department of Computer Engineering
Bilecik Seyh Edebali University
Turkiye
burakhan.cubukcu@bilecik.edu.tr

Ugur Yuzgec
Department of Computer Engineering
Bilecik Seyh Edebali University
Turkiye
ugur.yuzgec@bilecik.edu.tr

Abstract: *This study proposes constrained convolutional neural network models for determining the initial connection weights in the Fully Connected Network (FCN) layer within the Convolutional Neural Network (CNN) model, resulting in an increase in the CNN model's performance. A literature review indicates that the constrained method is used in conjunction with CNN. However, previous studies have typically focused on using the constrained method before feature selection in CNN. In contrast, this study aims to calculate the initial values of the connection weights, one of the hyperparameters in the FCN, by using the constrained method between feature selection and the FCN layer. Five different models are proposed: The constrained Difference CNN (D-CNN), the sample Constrained CNN (C-CNN), the constrained Sum CNN (S-CNN), the Random Sum CNN (RS-CNN), and the constrained Mixed CNN (M-CNN). These proposed models and classical CNN, have been applied to the Modified National Institute of Standards and Technology (MNIST), MNIST fashion, and CIFAR-10 datasets then the results have been examined. According to the average accuracy results, the C-CNN model achieved the highest performance in the MNIST dataset with an accuracy rate of 99.03%. In the MNIST fashion dataset, the best result was obtained by the D-CNN model with an accuracy rate of 91.80%. Similarly, the D-CNN model achieved the highest performance in the CIFAR-10 dataset with an accuracy rate of 71.44%. D-CNN and C-CNN models have outperformed the other proposed models and the classical CNN. The proposed D-CNN model, which achieved successful performance on the MNIST fashion and CIFAR-10 datasets, was compared with other recent studies in the literature. The reason for the better performance of D-CNN is considered to be their calculation based on the differential operation of two different classes*.

Keywords: *Artificial intelligence, image classification, constrained, convolutional neural network, hyperparameter, deep learning*.

Received April 23, 2025; accepted September 15, 2025
https://doi.org/10.34028/iajit/23/2/7

1. Introduction

Nowadays, with the rapid progress of technology, artificial intelligence applications are widely used in various areas of daily life such as cyber security [48], healthcare applications [12, 27], automobiles [43], robotics [42], agriculture [24], and education [10]. The foundation of this widespread use lies in the ability to develop high-performance applications in recent years [26]. While artificial intelligence applications were initially developed mainly with machine learning algorithms based on statistical foundations [35], nowadays, they are developed using neural network models such as Convolutional Neural Network (CNN) [32], Long Short-Term Memory (LSTM) [20], Extreme Learning Machine (ELM) [21], Multi-Layer Perceptron (MLP), Generative Adversarial Networks (GAN) [17], which constitute the concept of deep learning [31]. Particularly in deep learning studies related to image processing, the CNN model is frequently observed [25].

CNN is a deep learning model that has achieved significant success in image processing and pattern recognition, such as classifying images into categories [44]. First proposed by Lecun *et al*. [32] in 1998, this model has been used in various scientific studies and everyday applications with different usage methods and architectures from that time to the present. All architectures include some parameters. These parameters determined by the creators of the model are called hyperparameters [9]. These hyperparameters, initially assigned by the creator or randomly, affect the performance of the created model. There is no definitive method for selecting these hyperparameters, which are typically chosen randomly, to yield the best and fastest results in problems but there are different approaches on how parameters specific to the problem will be more suitable [40, 45, 52].

To choose the best hyperparameters for problems, some studies are based on trial and error, while others attempt to find optimal values with algorithmic support [53]. The trial-and-error-based simplest approach is named manual search [23]. In this approach, the person creating the model selects the parameters that give the

best results by trying different values [8]. In the grid search approach, which is another trial-and-error-based method, different combinations of values are automatically tested in a specific order. This method is quite inefficient in terms of time cost due to the large number of trials [6]. In the random search approach, the number of repetitions in grid search is reduced, and optimal parameters are sought by providing random values [7]. As alternatives to these trial-based methods, there are approaches for determining hyperparameters with metaheuristic algorithms [1, 16, 37]. While metaheuristic algorithms are commonly used for CNN [16, 38], the constrained weight/bias generation approach has been widely used in recent years for ELM.

Although various strategies exist for hyperparameter optimization, the initial weights and biases of the Fully Connected Network (FCN) layer within CNNs are often initialized randomly. This random initialization can sometimes lead to slower convergence or suboptimal performance. In this study, we propose a novel approach to address this by determining the initial connection weights in the FCN layer using constrained methods, aiming to enhance the overall performance of the CNN model.

2. Related Works

Before looking at specific studies that use constrained methods in CNN architectures, it is important to understand the broader context of hyperparameter optimization and the integration of structural constraints in deep learning models. While various strategies exist to optimize CNN performance, the direct incorporation of constraints into the network architecture, particularly for weight initialization, offers a distinct and promising path, including meta-heuristic approaches for hyperparameter tuning [1, 16, 37, 38]. Related studies have shown that jointly optimizing feature extraction mechanisms and network hyperparameters can significantly enhance learning stability and classification performance in CNN-based models [11]. The studies examined in this section focus on the application of constrained methods in CNNs and highlight their evolution and impact on performance, particularly in feature extraction and classification layers. This background provides the foundation for understanding the gap in the literature that our proposed approach aims to address.

In one notable study, Bayar and Stamm [5] proposed a constrained convolutional filter design for image manipulation detection, demonstrating the effectiveness of incorporating structural constraints into CNN architectures. For this detection, the constrained method was employed in the feature extraction section of CNN. A Constrained convolution layer was created, and after processing images in this layer, images were subjected to the classical convolution process. The results showed that while the standard CNN achieved an accuracy of 98.70%, the proposed constrained model improved the performance to 99.60% [5].

In a different study, the constrained Regional Convolution Neural Network (R-CNN) model was proposed. In this study, a similar approach to Bayar was taken by adding a new layer in front of the convolution layer. The proposed method was tested on the Common Objects in Context (COCO) synthetic dataset [51]. In the test results, constrained R-CNN obtained an F1 score of 0.927, while the compared Red-Green-Blue with Normal maps (RGB-N) and Multi-Feature Convolutional Network (MFCN) models achieved scores of 0.722 and 0.570, respectively.

The common feature of these studies is the use of the constrained method for the feature extraction process, which is the first part of the CNN architecture. In contrast to this architecture, there is a study, which the FCN layer within CNN was completely removed, and a constrained Extreme Learning Machine (ELM) classifier was added [39]. In their study where they performed scene classification on real images, they compared the CNN-ELM method with the proposed CNN-constrained ELM method. While CNN-constrained ELM achieved a performance of 98.10%, the CNN-ELM method achieved a performance of 91.83%.

In addition to different structure studies which is Zhu *et al*. [54] have proposed a model for ELM in their study titled “Constrained Extreme Learning Machines: A Study on Classification Cases” for Constrained CNN. In the proposed model, instead of randomly assigning initial parameters, they developed five different methods that would produce more suitable results by performing some mathematical operations on the data in the dataset. Thanks to five models Zhu’s *et al*. [54] proposed had an improvement accuracy of approximately 3% for the best proposed model.

As seen in the literature, it is observed that using constrained weight/bias generation structures with CNN increases the performance rate [34]. To increase the performance rate, the constrained structure is generally used before the feature extraction section [28]. Besides the feature extraction section, it is observed that the constrained ELM structure is used instead of the FCN layer. Using this constrained structure with CNN contributes to the literature [47]. However, to the best of our knowledge, no studies to date have focused on the constrained structure used between feature extraction and the FCN layer.

This study mainly focuses on determining the initial parameters of the FCN layer by putting the constrained weight generation structure between the feature extraction and FCN layer. In this new structure, five different constrained models inspired by Zhu’s *et al*. [54] study have been proposed. The proposed models are structurally similar to each other, but simple changes in the mathematical operations and algorithms used differentiate the models. The created models were

applied to the Modified National Institute of Standards and Technology (MNIST), MNIST fashion, and CIFAR10 datasets with fixed values such as batch size, epoch, and learning rate in classic CNN, and the results were compared.

In the continuous parts of this study, methods, results and discussion, and conclusion sections are presented, where the constrained CNN model, which is different from other studies, is applied to three different datasets and its outcomes are examined.

3. Methodology

CNN is a deep learning model used in image processing applications. This model consists of two main components. The first component is the feature extraction section where convolution operations are performed. The second section is the FCN where classification processes are carried out. While creating this architecture, the numbers and sequence of layers, filter sizes, neuron numbers, and many other parameters are mostly determined by the model creator.

One of these parameters which are mostly determined randomly is the initial values of the connection weights and biases in the FCN layer. In this study, the main goal is to improve the performance of the original model by determining the initial values of the connection weights/biases in the FCN layer using the constrained weight generation method. The constrained weight generation methods architecture has been created in two steps to enable constrained calculations. In Figure 1, the section featuring constrained weight generation calculations is presented as step-1, whereas the portion where the results obtained from the constrained weight generation computation are applied to the CNN model is presented as step-2.

At the beginning of step 1, there is the feature extraction section of CNN. Following this section, step 1 is completed with the section where constrained weight generation calculations are performed. All data pass through the first part which includes feature extraction with a single batch, resulting in obtaining vectors for each image in the flatten layer. After that two random vectors among these created vectors are selected, and in the next section, constrained calculations, which will be detailed later, are performed to create the first row of the vector connection weights matrix. To determine the values of all connection weights, this process was repeated for the number of neurons. When repeated as many times as the number of neurons, the obtained matrix constitutes the connection weights of the network in the FCN layer. With the acquisition of initial parameters in the FCN layer, step 1 is completed, and step 2 is initiated.

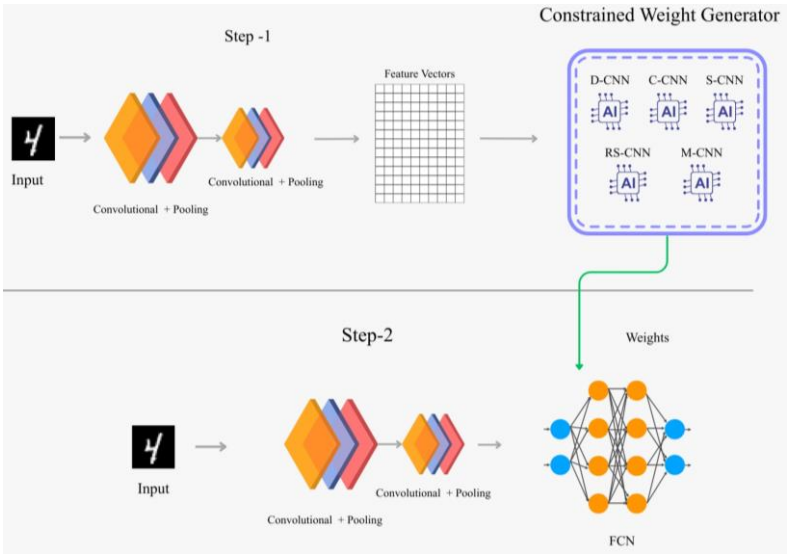


Figure 1. Constrained CNN model architecture.

Step-2 consists of two parts: the first part, where the feature extraction processes of the CNN model are performed which are similar to step-1, and the second part, which includes the classification processes in the FCN layer. While creating the first part, all parameters such as the number of convolution layers, stride value, pooling operations, etc., are kept the same as in step 1. steps 1 and 2 keep the parameters in the feature extraction section constant, resulting in the same vectors being obtained. When forming the second part of step-2, the values obtained from the constrained weight generation calculation in step 1 are used as the initial parameters of weights in the FCN layer.

During the model creation, two-layered convolution was applied in both steps 1 and 2. In both steps convolution layers, stride, and padding values were set to 1, and a 3x3 kernel was utilized. After each convolution layer, max-pooling was implemented. Later than the pooling layers, FCN layer, created in step-2, an MLP with one hidden layer was established. This MLP has 128 neurons in the input layer and 10 neurons in the output layer. This generated model is used for three datasets.

In the following method section, step-by-step

algorithms are provided for how each of the five proposed constrained methods is calculated, and the differences between them are explained. After that the datasets used and the accuracy, loss, F1-score, precision, recall metrics employed for performance measurement are included.

3.1. Constrained Convolutional Neural Network

A constrained weight generator allows obtaining an average value through mathematical operations based on the relationships between classes in the dataset without undergoing a training process. In this study, a constrained weight generator has been utilized to determine the initial parameters of connection weights in the FCN layer within CNN. Five different constrained methods were employed in the study. The detailed operations and algorithms of these methods are provided in the continuation of this section.

3.1.1. Constrained Difference Convolutional Neural Network (D-CNN)

In the D-CNN method, computational operations are performed on the vectors obtained after the images are processed through the convolution operation. These computations involve taking half of the vector differences. This process of taking half of the vector differences is repeated for each neuron in the FCN layer of the constructed model. In this study, since the FCN layer of the model uses 128 neurons, the computation process is repeated 128 times. The weights obtained for these 128 neurons constitute the initial values of the weights in the FCN layer. The pseudo code of the D-CNN method that enables obtaining these initial values is provided in Algorithm (1).

Algorithm 1. Pseudo code of D-CNN model.

1) *The dataset undergoes a convolution operation, and the data from the flatten layer is obtained.*
2) *for i=1:N do→N: the number of neurons in the FCN input layer*
   a) *Randomly select two data points X_{c1} and X_{c2} from any two different classes*
   b) *Generate the difference vector X_{c2}-X_{c1}*
   c) *Normalize the difference vector by $w_i=\frac{2(X_{c2}-X_{c1})}{||X_{c2}-X_{c1}||_{L_2}^2}$*
   d) *Calculate the bias $b_i=\frac{(X_{c1}+X_{c2})^T(X_{c1}-X_{c2})}{||X_{c2}-X_{c1}||_{L_2}^2}$*
   e) *Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i*
3) *end for*
4) *Assign the obtained values W and b as initial values of the FCN part of the CNN model.*
5) *Start the training process of CNN model.*

3.1.2. Sample Constrained Convolutional Neural Network (C-CNN)

In this sample constrained CNN (C-CNN) model, as in the previous model, the initial values of the weights and biases of the FCN in the last layer of the CNN model are assigned depending on the data set. The pseudo code for this model is given in Algorithm (2). Initially, the dataset undergoes a convolution operation, resulting in data from the flatten layer.

Then, a loop iterates N times, representing the number of neurons in the FCN input layer. Within each iteration, the random data points x_{c1} and x_{c2} are chosen from the same classes, and the difference of these points is calculated and the difference vector is normalized using the L_2 norm. Subsequently, a bias b_i is randomly assigned from the interval [0, 1]. With the weights utilizing the normalized difference vector and the randomly determined bias, the algorithm computes the corresponding rows of the weight matrix $W_{i\times L}$ and bias vector $b_{i\times L}$.

Finally, the obtained weight matrix W and bias vector b are set as the initial values for the FCN part of the CNN model, initiating the training process for the CNN model.

Algorithm 2. Pseudo code of C-CNN model.

1) *The dataset undergoes a convolution operation, and the data from the flatten layer is obtained.*
2) *for i=1:N do→N: the number of neurons in the FCN input layer*
   a) *Randomly select two data points X_{c1} and X_{c2} from the same classes*
   b) *Generate the difference vector X_{c2}-X_{c1}*
   c) *Normalize the difference vector by $w_i=\frac{2(X_{c2}-X_{c1})}{||X_{c2}-X_{c1}||_{L_2}^2}$*
   d) *Determine the bias b_i randomly from [0, 1]*
   e) *Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i*
3) *end for*
4) *Assign the obtained values W and b as initial values of the FCN part of the CNN model.*
5) *Start the training process of CNN model.*

3.1.3. Constrained Sum Convolutional Neural Network (S-CNN)

S-CNN utilizes sum vectors of randomly selected classes to construct the weights from the input layer to the hidden layer. The S-CNN algorithm randomly selects two class sample vectors (x_{c1}, x_{c2}) calculates their sum, and normalizes the resulting sum vector. These normalized sum sample vectors are then assigned as the weights from the input layer to the hidden layer. Additionally, the biases used in this model are randomly generated from the uniform distribution. The method is presented in the following Algorithm (3).

Algorithm 3. Pseudo code of S-CNN model.

1) *The dataset undergoes a convolution operation, and the data from the flatten layer is obtained.*
2) *for i=1:N do→N: the number of neurons in the FCN input layer*
   a) *Randomly select two data points X_{c1} and X_{c2} from the same classes*
   b) *Generate the sum vector X_{c2}+X_{c1}*
   c) *Normalize the sum vector by $w_i=\frac{(X_{c2}+X_{c1})}{||X_{c2}-X_{c1}||_{L_2}^2}$*
   d) *Determine the bias b_i randomly from [0, 1]*

e) Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i
3) end for
4) Assign the obtained values W and b as initial values of the FCN part of the CNN model.
5) Start the training process of CNN model.

Looking at this pseudo code, the most important difference of this model from the previous two models (D-CNN and C-CNN) is that it uses the sum of the samples of two randomly selected classes to assign the initial values of the weights in the FCN part. These classes are selected so that their labels are the same. That is, two different instances of the same class are selected. The bias values of the FCN are randomly selected and assigned as in the C-CNN model. All these steps continue until weights and bias values are generated for all hidden neurons of the FCN. The final weights and bias values are assigned as the initial values of the FCN part of the CNN model and the training process is started.

3.1.4. Random Sum Constrained Convolutional Neural Network (RS-CNN)

RS-CNN utilizes sum vectors of randomly sampled vectors to generate weights from the input layer to the hidden layer of the FCN part. First, the model randomly selects two data points from the same classes, then collects the sample values of these points for the sum vector. This sum vector is normalized to find the initial weights in the FCN part of the CNN model. The bias values in the FCN are also randomized between 0 and 1 in this model. The pseudo code of the RS-CNN model is given in Algorithm (4).

Algorithm 4. Pseudo code of RS-CNN model.

1) The dataset undergoes a convolution operation, and the data from the flatten layer is obtained.
2) for i=1:N do→N: the number of neurons in the FCN input layer
a) Randomly select two data points X_{c1} and X_{c2}
b) Generate the sum vector $X_{c2}+X_{c1}$
c) Normalize the sum vector by $w_i=\frac{(X_{c2}+X_{c1})}{||X_{c2}+X_{c1}||_{L_2}^2}$
d) Determine the bias b_i randomly from [0, 1]
e) Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i.
3) end for
4) Assign the obtained values W and b as initial values of the FCN part of the CNN model.
5) Start the training process of CNN model.

3.1.5. Constrained Mixed Convolutional Neural Network (M-CNN)

This model is created by combining the modified D-CNN and RS-CNN models. The operations of the models are repeated as many times as the number of neurons. In this method, however, the number of neurons is divided in half, applying RS-CNN for the first half and the modified D-CNN for the second half, resulting in a mixed approach. M-CNN is presented in the following Algorithm (5). Unlike the normal D-CNN model, this modified D-CNN model generates randomly the bias values in the FCN part of the CNN model.

Algorithm 5. Pseudo code of M-CNN model.

1) The dataset undergoes a convolution operation, and the data from the flatten layer is obtained.
2) for i=1:N do→N: the number of neurons in the FCN input layer
a) if i<(N/2)+1 then
1. Randomly select two data points X_{c1} and X_{c2} from the same class
2. Generate the sum vector $X_{c2}+X_{c1}$
3. Normalize the sum vector by $w_i=\frac{(X_{c2}+X_{c1})}{||X_{c2}+X_{c1}||_{L_2}^2}$
4. Determine the bias b_i randomly from [0, 1]
5. Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i.
b) else
1. Randomly select two data points X_{c1} and X_{c2} from any two different classes
2. Generate the difference vector X_{c2}-X_{c1}
3. Normalize the difference vector by $w_i=\frac{2(X_{c2}-X_{c1})}{||X_{c2}-X_{c1}||_{L_2}^2}$
4. Determine the bias b_i randomly from [0, 1]
5. Obtain the ith row of the matrix $W_{i\times L}$ and bias vector $b_{i\times L}$ using the w_i and b_i.
c) endif
3) end for
4) Assign the obtained values W and b as initial values of the FCN part of the CNN model.
5) Start the training process of CNN model.

3.2. Data Sets

In this study, commonly found datasets in the literature, namely MNIST, MNIST fashion, and CIFAR10, were used. These datasets were selected to avoid excessive memory usage, as the images in these datasets are small in size.

3.2.1. MNIST Dataset

The MNIST dataset, which contains handwritten digits, was created in 1994 by combining datasets from the National Institute of Standards and Technology (NIST) [30]. It has become one of the most frequently used datasets in image processing studies and has appeared in numerous works in the literature. Figure 2 presents sample images of the MNIST dataset.

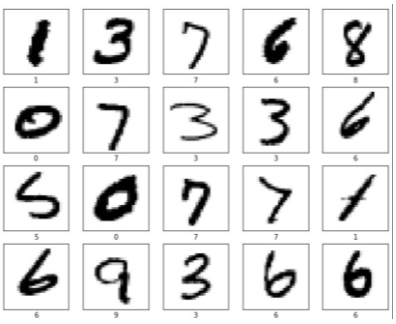


Figure 2. MNIST data examples.

The MNIST dataset consists of 70,000 grayscale images of handwritten digits, each measuring 28x28 pixels, and includes 10 classes covering digits 0 to 9. Its balanced distribution of classes, small image size, and numerous comparative studies in the literature make MNIST an especially useful dataset [4].

3.2.2. MNIST Fashion Dataset

Introduced as an alternative to the classic MNIST dataset in 2017, the MNIST Fashion dataset includes images of fashion items. This dataset was created to make models struggle more than MNIST, with high accuracy values. As depicted in Figure 3, images in the MNIST Fashion dataset are saved in black and white, with dimensions of 28x28 pixels [49].

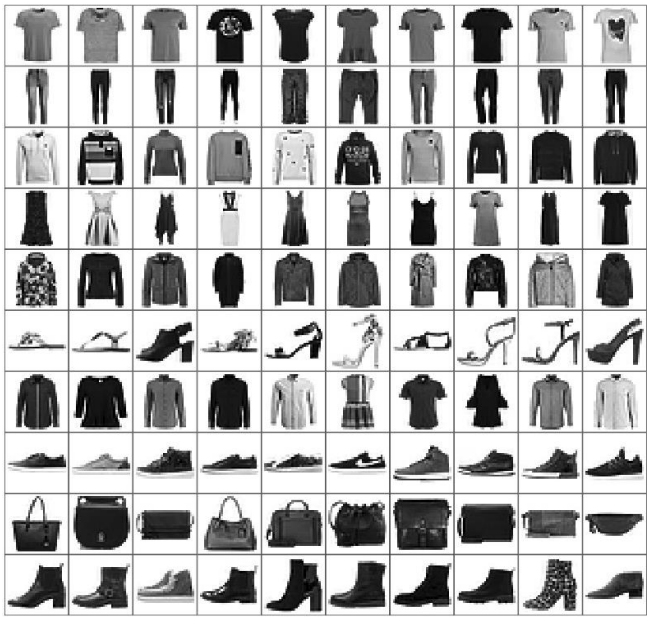


Figure 3. MNIST Fashion data examples.

Similar to the MNIST dataset, it is divided into 10 classes, with 70,000 images. Of these, 60,000 images are allocated for training, and 10,000 for testing. The dataset exhibits an equal distribution of images for each class, ensuring a balanced representation.

3.2.3. CIFAR10 Dataset

The CIFAR-10 dataset, as shown in Figure 4, was collected and labeled at the University of Toronto in 2008 by Alex Krizhevsky and colleagues, and it was made publicly accessible in 2009 [29]. This dataset is widely used in the literature and consists of 32x32 RGB images featuring animals and vehicles. While the other two datasets used in this study include grayscale images, CIFAR-10 is unique in that it contains RGB images.

The CIFAR-10 dataset consists of 60,000 images, with 50,000 allocated for training and 10,000 for testing. Similar to the other datasets used in this study, it includes data for a total of 10 different classes, with 6,000 images per class, contributing to a well-balanced distribution of data.

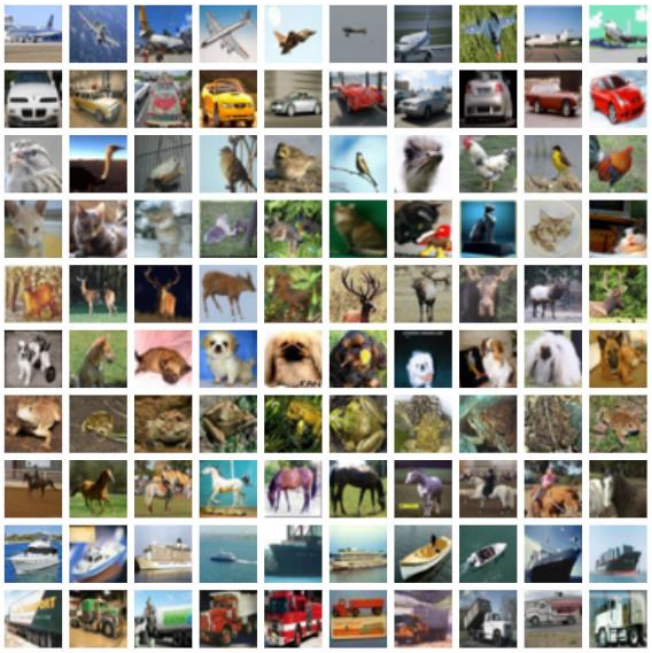


Figure 4. CIFAR10 data examples.

It is important to note that the datasets used in this study (MNIST, MNIST fashion, and CIFAR-10) are characterized by a balanced class distribution, where each class contains an equal or nearly equal number of samples. Consequently, the issue of class imbalance was not a primary concern in our experimental setup. While this work focuses on evaluating the proposed constrained models under these balanced conditions, addressing class imbalance is a recognized challenge in machine learning. Common strategies for handling imbalanced data include resampling techniques (e.g., Synthetic Minority Over-sampling Technique (SMOTE) for oversampling), cost-sensitive learning (e.g., assigning higher penalties to misclassifications of minority classes), and the use of ensemble methods. Investigating the performance and adaptation of the proposed constrained weight initialization method in scenarios with significant class imbalance is an interesting direction for future research.

4. Results and Discussion

In this section of the study, the proposed models outlined in the methodology section were applied to the MNIST, MNIST Fashion, and CIFAR-10 datasets, and the results were analyzed. In the continuous of the section, the proposed model is compared with current studies in the literature.

4.1. Evaluation Metrics

To ensure clarity and reproducibility, this section provides the definitions and mathematical formulations for the performance evaluation metrics (accuracy, precision, recall, F1-score, and loss) used in this study. These metrics were calculated based on the confusion matrix, which summarizes the predictions made by the

model, distinguishing between True Positives (*TP*), True Negatives (*TN*), False Positives (*FP*), and False Negatives (*FN*) for each class. Using these values, Accuracy (*Acc*), Precision (*P*), Recall (*R*), F1-score (*F*1), and Loss (*L*) are calculated as presented in Equations (1), (2), (3), (4), and (5).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$P = \frac{TP}{TP + FP} \quad (2)$$

$$R = \frac{TP}{TP + TN} \quad (3)$$

$$F1 = 2 \times \frac{P \times R}{P + R} \quad (4)$$

$$L = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \quad (5)$$

Here, C is the number of classes, y_i is the true label (0 or 1) for class i, and $\hat{y}_i$ is the predicted probability for class i.

4.2. Comparison of the Proposed Models

To compare the proposed models, the classic CNN model was also trained on the same datasets. All models were trained five times on each dataset, and the results of the tests conducted after training were presented in tables for the best, worst, and average outcomes. For all datasets used in the study, the classic CNN model and the proposed C-CNN, S-CNN, D-CNN, M-CNN, and RS-CNN models were applied. Firstly, models were tested on the MNIST dataset, and the results of the evaluation, including the best, worst, and mean outcomes, are presented in Table 1.

Table 1. Comparison results of constrained CNNs and classic CNN for MNIST dataset.

	Metrics	Accuracy	Loss	F1-score	Precision	Recall
CNN	Worst	98.52	0.0558	0.9851	0.9853	0.9850
	Mean	98.62	0.0511	0.9862	0.9863	0.9861
	Best	98.92	0.0403	0.9891	0.9892	0.9888
C-CNN	Worst	98.94	0.0388	0.9892	0.9894	0.9890
	Mean	99.03	0.0370	0.9903	0.9901	0.9902
	Best	99.16	0.0323	0.9917	0.9914	0.9919
S-CNN	Worst	98.33	0.0554	0.9833	0.9834	0.9832
	Mean	98.55	0.0469	0.9856	0.9856	0.9855
	Best	98.73	0.0393	0.9874	0.9874	0.9875
D-CNN	Worst	98.73	0.0457	0.9874	0.9873	0.9874
	Mean	98.92	0.0412	0.9892	0.9890	0.9894
	Best	99.06	0.0402	0.9907	0.9906	0.9907
M-CNN	Worst	98.47	0.0478	0.9848	0.9848	0.9847
	Mean	98.57	0.0426	0.9858	0.9859	0.9857
	Best	98.89	0.0342	0.9888	0.9886	0.9889
RS-CNN	Worst	98.37	0.0570	0.9835	0.9836	0.9834
	Mean	98.43	0.0539	0.9845	0.9844	0.9845
	Best	98.76	0.0433	0.9878	0.9877	0.9879

In this comparison study, accuracy, loss, F1-score, precision, recall metrics were used to evaluate the performance of the models. For the model showing the best performance, confusion matrices and Receiver Operating Characteristic (ROC) curves were plotted for each dataset. As can be seen from Table 1, the classical CNN is compared with the proposed models. According to this comparison, C-CNN, D-CNN, and S-CNN demonstrate better performance, while M-CNN and RS-CNN show lower performance than classic CNN. The C-CNN model obtained the highest accuracy and the best average accuracy values. The confusion matrix of the C-CNN model achieving the best results is presented in Figure 5 and the Receiver Operating Characteristic-Area Under the Curve (ROC-AUC) curve is shown in Figure 6.

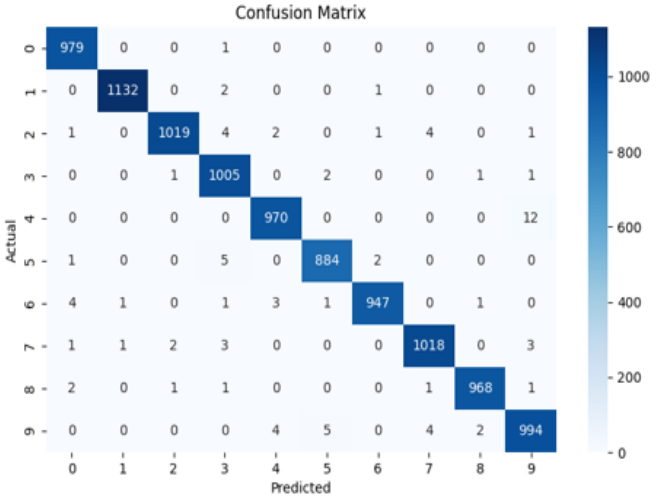


Figure 5. Confusion matrix of the C-CNN model for the MNIST dataset.

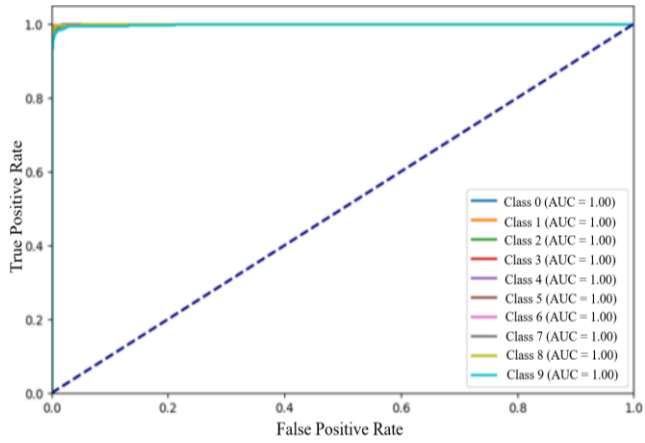


Figure 6. AUC-ROC graph of the C-CNN model for the MNIST dataset.

As presented in Figure 5, the most successfully predicted class is "1," while the least successful is class "4." As illustrated in the ROC-AUC curve, it can be observed that the success percentages of classes with very similar results are close to each other.

According to the results, especially accuracy, there is a difference of 0.60% between the best result, C-CNN, with 99.03% accuracy, and the worst average accuracy, RS-CNN, with 98.43% success. The close proximity of all metric results makes it difficult to compare the models. To facilitate this comparison, the models were tested on two different, more complex datasets than the MNIST dataset.

The MNIST Fashion dataset was created to increase

the difficulty of the problem compared to the MNIST dataset and to further challenge the models. Since close results were obtained in this study with the MNIST dataset, the more complex MNIST Fashion dataset was used. The results obtained for the MNIST Fashion dataset are presented in Table 2.

Table 2. Comparison results of constrained CNNs and classic CNN for MNIST fashion dataset.

	Metrics	Accuracy	Loss	F1-score	Precision	Recall
CNN	Worst	89.91	0.296	0.898	0.900	0.897
	Mean	89.95	0.291	0.899	0.900	0.899
	Best	89.97	0.289	0.900	0.901	0.899
C-CNN	Worst	91.39	0.256	0.913	0.914	0.914
	Mean	91.41	0.255	0.914	0.915	0.915
	Best	91.42	0.254	0.915	0.916	0.915
S-CNN	Worst	89.31	0.294	0.894	0.895	0.894
	Mean	89.69	0.293	0.899	0.900	0.897
	Best	90.03	0.291	0.900	0.903	0.900
D-CNN	Worst	91.51	0.258	0.916	0.916	0.915
	Mean	91.80	0.253	0.918	0.917	0.918
	Best	92.11	0.251	0.921	0.922	0.921
M-CNN	Worst	90.87	0.259	0.908	0.909	0.908
	Mean	90.92	0.256	0.909	0.910	0.909
	Best	90.94	0.253	0.910	0.911	0.910
RS-CNN	Worst	88.31	0.339	0.885	0.887	0.884
	Mean	88.39	0.335	0.887	0.889	0.885
	Best	88.44	0.329	0.908	0.909	0.907

As demonstrated in Table 2, the classical CNN is compared with the proposed models. According to this comparison, C-CNN, D-CNN, and M-CNN demonstrate better performance, while S-CNN and RS-CNN show lower performance than classic CNN. M-CNN performs worse than classical CNN in the MNIST dataset although M-CNN is more successful in MNIST fashion which is a more challenging problem. While the worst results are obtained by RS-CNN on the best and mean accuracy, D-CNN achieves the best. D-CNN's confusion matrix is presented in Figure 7, and the ROC-AUC curve is shown in Figure 8.

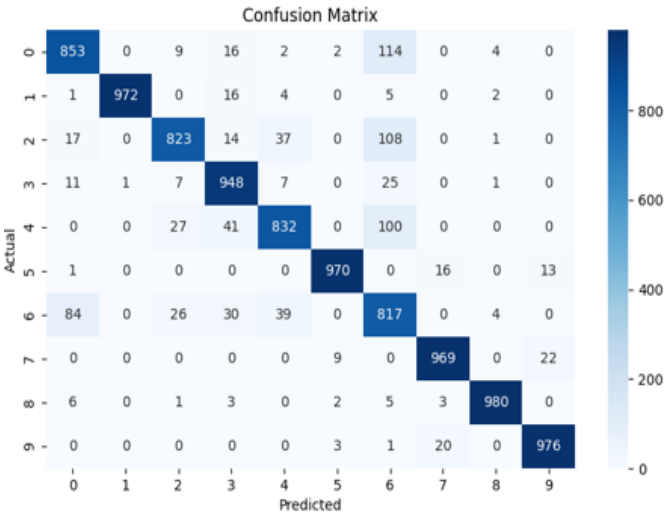


Figure 7. Confusion matrix of the D-CNN model for the MNIST fashion dataset.

In the confusion matrix, it is observed that the most successfully predicted class is the "1" class containing trouser images, and the least successfully predicted class is the "6" class which shirt images. Examining the ROC-AUC curve (Figure 8) reveals a clearer distribution compared to the MNIST dataset, enabling a more distinct differentiation of class performances.

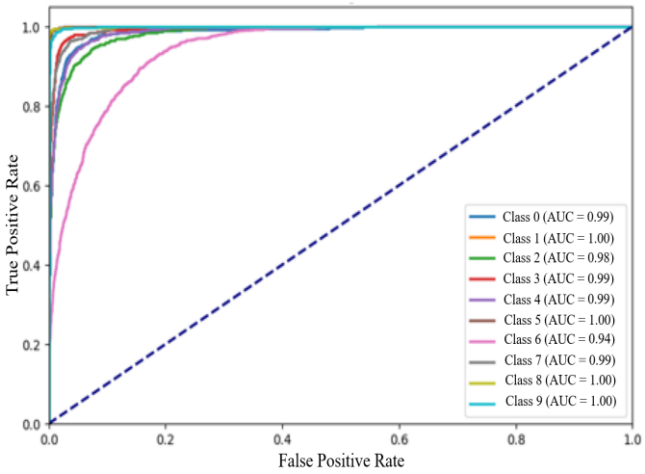


Figure 8. AUC-ROC graph of the D-CNN model for the MNIST fashion dataset.

When considering the overall distribution of performances, it is evident that D-CNN achieved the highest accuracy at 91.8%, while RS-CNN yielded the lowest outcome at 88.39%. There exists a 3.41% disparity between the highest and lowest average accuracy values. The difference between the best average and that of the classical CNN increased from 0.41% in MNIST to 3.41% in MNIST fashion, indicating clearer performance distinctions compared to the MNIST dataset. After the MNIST fashion dataset, the proposed models were lastly applied to the CIFAR-10 dataset. The CIFAR-10 dataset's results are presented in Table 3.

Table 3. Comparison results of constrained CNNs and classic CNN for CIFAR-10 dataset.

	Metrics	Accuracy	Loss	F1-score	Precision	Recall
CNN	**Worst**	65.14	1014	0.653	0.658	0.652
	Mean	65.59	1011	0.667	0.660	0.658
	Best	66.02	1009	0.669	0.662	0.660
C-CNN	**Worst**	71.01	0.990	0.711	0.718	0.710
	Mean	71.22	0.983	0.713	0.719	0.712
	Best	71.43	0.980	0.715	0.720	0.714
S-CNN	**Worst**	67.21	1046	0.672	0.680	0.672
	Mean	67.71	1039	0.677	0.686	0.676
	Best	67.98	1032	0.681	0.692	0.680
D-CNN	**Worst**	71.12	0.990	0.708	0.712	0.712
	Mean	71.44	0.983	0.713	0.720	0.714
	Best	71.69	0.980	0.719	0.728	0.717
M-CNN	**Worst**	68.69	1002	0.688	0.697	0.687
	Mean	68.94	0.994	0.690	0.703	0.689
	Best	69.19	0.992	0.695	0.709	0.692
RS-CNN	**Worst**	69.18	0.999	0.691	0.699	0.692
	Mean	69.97	0.981	0.701	0.712	0.709
	Best	70.55	0.963	0.711	0.717	0.712

CIFAR-10 contains RGB images on contrary to the other two datasets which consist of grayscale images. As a result of using RGB images, the flatten layer, formed as the input to the FCN layer, was created with 3 channels for this dataset.

According to the findings presented in Table 3, all proposed models have demonstrated superior performance compared to the classical CNN. Similar to the MNIST fashion dataset, the D-CNN model has emerged as the top performer. Conversely, the S-CNN model yielded the weakest results. The confusion matrix for the D-CNN model, which achieved the highest performance, is presented in Figure 9, and the ROC-AUC curve is depicted in Figure 10.

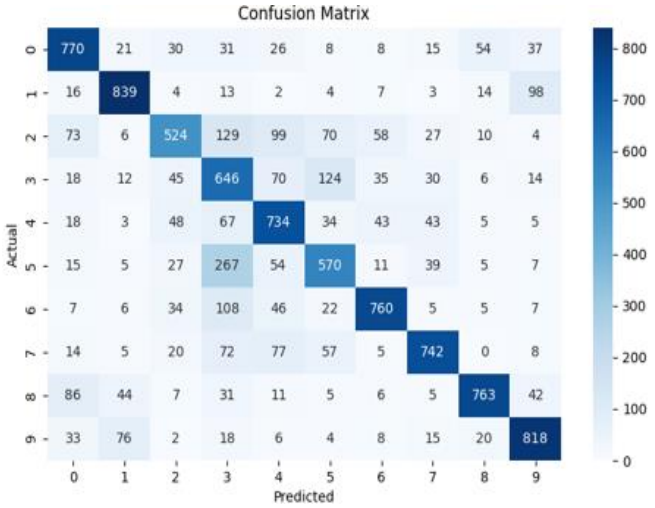


Figure 9. Confusion matrix of the D-CNN model for the CIFAR-10 dataset.

In the confusion matrix depicted in Figure 9, the category comprising car images is identified as the most accurately predicted class, whereas the category containing cat images is observed as the least accurately predicted class. After analysing the ROC-AUC curve shown in Figure 10, it is evident that the performance differences between classes have increased compared to the other two datasets. This allows for a clearer representation of performance differences among the models.

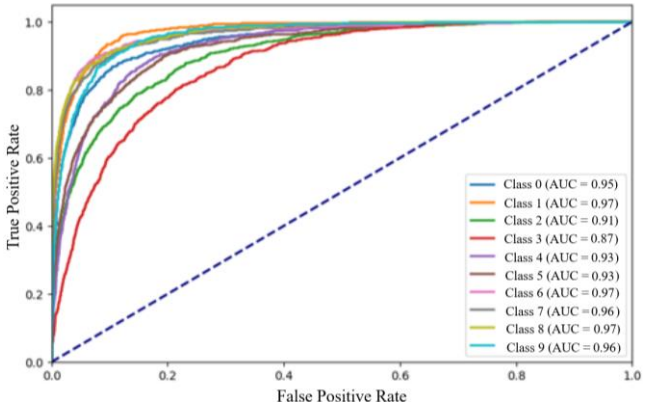


Figure 10. AUC-ROC graph of the D-CNN model for the CIFAR 10 dataset.

Following the evaluation of the CIFAR-10 dataset results are examined, as opposed to the other datasets, all the proposed models have outperformed the classical CNN. The D-CNN model, which achieved the highest result, obtained a mean accuracy that is 5.86% higher than the classical CNN, which achieved the lowest result.

When considering all results, it is observed that as the complexity of the problem increases, the proposed models achieve higher performances compared to the classical CNN. While the difference between the most successful model and the original CNN model is 0.63% for the MNIST, this difference increases to 1.85% for the MNIST Fashion and 5.86% for the CIFAR-10. The growth of this difference seems to be related to the increased difficulty of the problem. To better observe this difference, the average accuracy results obtained from the three datasets are visualized in Figure 11.

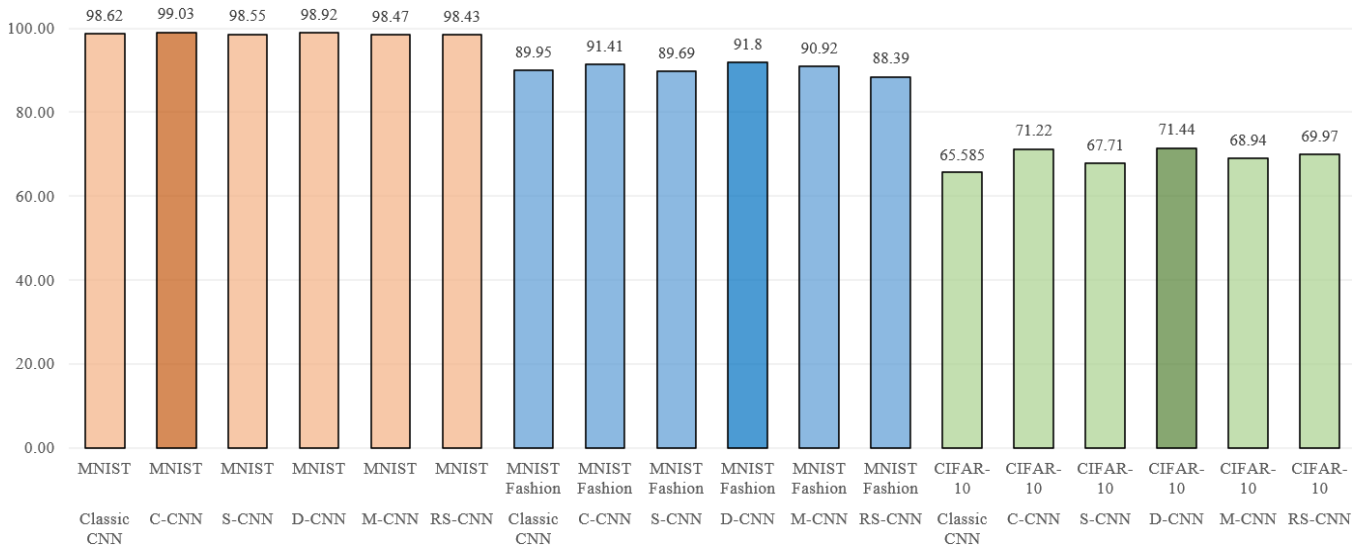


Figure 11. Accuracy results of the models for all datasets.

The top two performers across all datasets are C-CNN and D-CNN. Due to both of these models being based on differential operations, it is considered more beneficial to use these differential operation-based constrained models within CNNs. The performance of these two models suggests their potential as alternatives to classical CNNs.

In this study, the training times for the three datasets

were recorded. In Table 4, the average training times from five training sessions are given in seconds.

According to Table 4, the model with the lowest average was M-CNN. The training time with the best value belonged to Classic CNN. Among the proposed models, D-CNN achieved the best results for the MNIST and CIFAR-10 datasets, while C-CNN reached the top for the MNIST fashion dataset. When comparing the best values of the proposed models with classic CNN, there is approximately a 10% to 15% loss of time.

Table 4. Comparison of training times (seconds).

	Model					
Dataset	**D-CNN**	**C-CNN**	**M-CNN**	**SCNN**	**RSCNN**	**CNN**
MNIST	853	858	904	863	897	728
MNIST fashion	973	945	979	942	953	834
CIFAR-10	1123	1111	1162	1106	1125	943

4.3. Comparison of Literature Studies with D-CNN Model

The most successful D-CNN model among the proposed models has been compared with studies in the literature, particularly those conducted in recent years, based on accuracy metric. As the results obtained on the MNIST dataset are very close to each other, a detailed comparison has not been conducted. Initially, the D-CNN model that achieved the best results on the MNIST Fashion dataset is compared with the other studies in Table 5.

Table 5. Comparison results of D-CNN and literature studies for MNIST fashion dataset.

Year	**Study**	**Model**	**Accuracy (%)**
2024	Venkataravanappa *et al.* [46]	Resnet50	90.25
2023	Xin *et al.* [50]	Tiny VGG (max pooling)	88.21
		(min pooling)	87.78
2024	Ortiz *et al.* [36]	ESN	88.10
		MR ESN	89.12
2021	Huang *et al.* [22]	Laser based RC	85.46
2023	Haider *et al.* [19]	COVNet	88.40
2023	Sun *et al.* [41]	MADPL-Net	90.11
2023	Erkoc and Eskil [15]	Unsupervised filter learning method	91.24
2025	**Our Proposed Model**	**D-CNN**	**91.80**

Table 6. Comparison results of D-CNN and literature studies for CIFAR-10 dataset.

Year	**Study**	**Model**	**Accuracy (%)**
2024	Gronningsaeter *et al.* [18]	CTM	60.70
2024	Azam and Akhtar [3]	Simple MLP	39.50
		ConNet(Large)	71.00
		ConvKANLinear	61.60
		KConvKAN	62.60
2024	Li *et al.* [33]	LENET-5	55.16
		ALEXNET	69.75
2023	Dogan [14]	Lenet-5 2×2 AVG pooling	66.95
2023	Aslam and Nassif [2]	VGG-16	61.00
		VGG-19	60.00
2023	Cui *et al.* [13]	Tesla (10 Epoch)	66.40
		FrePro (10 Epoch)	65.50
		MTT (10 Epoch)	65.30
2025	**Our Proposed Model**	**D-CNN**	**71.44**

The proposed D-CNN model was compared to recent studies in the literature using the accuracy metric on the MNIST Fashion dataset. It was observed that the D-CNN achieved a strong performance among the recent studies. While there were studies yielding similar results, the D-CNN model delivered the best performance. Following this comparison, the proposed model was also evaluated on another commonly used dataset in the literature, the CIFAR-10 dataset, and the results are presented in Table 6.

This improvement, particularly at lower epoch values, can be attributed to the starting value of connection weights in the FCN layer, as opposed to being randomly initialized, aligning with the objective of our study.

5. Conclusions

In this study, five different constrained CNN models named constrained D-CNN, sample C-CNN, constrained S-CNN, RS-CNN, and M-CNN have been proposed to determine the initial parameters of connection weights in the FCN layer of the CNN model with the aim of improving the performance of the CNN model. In all these proposed constrained CNN models, all data is passed through the convolution process, constrained weight generation calculations are performed to determine the initial parameters, and then CNN training process is started using these specified initial parameters. Tests are conducted after training, and the results are compared based on accuracy, precision, recall, and F1-score metrics.

According to the results obtained from testing the proposed models, different models perform the best in each dataset, and some of the proposed models achieve higher performance than the classical CNN method. In the MNIST dataset, where the Classical CNN model achieves an average accuracy of 98.68%, the proposed C-CNN model has the highest performance among the compared models with a 99.03% accuracy. The performance values in this dataset are very close to each other, and there is no significant difference. To better understand the performance of the proposed models, the models were also compared on the more challenging MNIST Fashion and CIFAR-10 datasets. In the MNIST Fashion dataset, where the Classical CNN model achieves a performance of 89.95%, the D-CNN model achieves the highest performance with 91.8%. Finally, in the CIFAR-10 dataset, the Classical CNN achieves a performance of 65.58%, while the D-CNN model achieves a performance of 71.44%. The comparisons made on these two datasets have facilitated a better evaluation of the performance of the five proposed models.

D-CNN model that achieved the best results was compared with recent studies in the literature for the MNIST Fashion and CIFAR-10 datasets. It was compared with seven studies for the MNIST Fashion dataset and six studies for the CIFAR-10 dataset. Specifically, in a study by Cui *et al.* [13] for CIFAR-10,

where the number of epochs was the same, an accuracy of 66.4% was achieved, while the D-CNN model showed a 5.4% improvement, reaching an accuracy of 71.44%. The initialization of connection weights in the FCN layer with a well-defined value allowed the proposed models to stand out, particularly during a short training period of just 10 epochs, reflecting the expected outcome of the study.

When evaluating the achievements of the proposed five models, C-CNN and D-CNN generally provide close and high results. The reason behind the high achievements and close results of these algorithms is observed to be their similarity in algorithms and their utilization of subtraction operations in both cases. As the problem complexity increases, it is observed that the performance difference between proposed models and classical CNN increases even further. It is thought that the reason for constrained weight generate methods outperforming classical CNN, especially as the problem becomes more challenging, is constrained weight generator methods' ability to calculate based on the relationships within complex data distributions.

In the future, to improve the computation method when training on datasets containing more complex data distributions, heuristic algorithms could be used to optimize the selection of randomly chosen classes during computation.

Acknowledgment

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. While language and grammar editing assistance was provided by the Qwen AI model, the authors retain full responsibility for the content, scientific accuracy, and integrity of the work.

References

[1] Alsubai S., Alqahtani A., and Sha M., "Genetic Hyperparameter Optimization with Modified Scalable-Neighbourhood Component Analysis for Breast Cancer Prognostication," *Neural Networks*, vol. 162, pp. 240-257, 2023. https://doi.org/10.1016/j.neunet.2023.02.035

[2] Aslam S. and Nassif A., "Deep Learning Based CIFAR-10 Classification," *in Proceedings of the Advances in Science and Engineering Technology International Conferences*, Dubai, pp. 1-4, 2023. https://doi.org/10.1109/ASET56582.2023.10180767

[3] Azam B. and Akhtar N., "Suitability of KANs for Computer Vision: A Preliminary Investigation," *arXiv Preprint*, vol. arXiv: 2406.09087v2, pp. 1-11, 2024. https://arxiv.org/abs/2406.09087v2

[4] Baldominos A., Saez Y., and Isasi P., "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, pp. 1-16, 2019. https://doi.org/10.3390/app9153169

[5] Bayar B. and Stamm M., "Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691-2706, 2018. https://doi.org/10.1109/TIFS.2018.2825953

[6] Belete D. and Huchaiah M., "Grid Search in Hyperparameter Optimization of Machine Learning Models for Prediction of HIV/AIDS Test Results," *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 875-886, 2022. https://doi.org/10.1080/1206212X.2021.1974663

[7] Bergstra J. and Bengio Y., "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, pp. 281-305, 2012. https://jmlr.org/papers/v13/bergstra12a.html

[8] Bergstra J., Bardenet R., Bengio Y., and Kegl B., "Algorithms for Hyper-Parameter Optimization," *in Proceedings of the 25th International Conference on Neural Information Processing Systems*, New York, pp. 2546-2554, 2011. https://dl.acm.org/doi/10.5555/2986459.2986743

[9] Cabada R., Rangel H., Estrada M., and Lopez H., "Hyperparameter Optimization in CNN for Learning-Centered Emotion Recognition for Intelligent Tutoring Systems," *Soft Computing*, vol. 24, no. 10, pp. 7593-7602, 2020. https://doi.org/10.1007/s00500-019-04387-4

[10] Chen L., Chen P., and Lin Z., "Artificial Intelligence in Education: A Review," *IEEE Access*, vol. 8, pp. 75264-75278, 2020. https://doi.org/10.1109/ACCESS.2020.2988510

[11] Coronel L., Fajardo A., and Medina R., "Horizontal Sequence Pooling Technique in Convolutional Neural Networks to Optimize Feature Extraction for DNA Sequence Classification," *The International Arab Journal of Information Technology*, vol. 21, no. 5, pp. 844-853, 2024. https://doi.org/10.34028/iajit/21/5/6

[12] Cubukcu B., Yuzgec U., Zileli R., and Zileli A., "Reliability and Validity Analyzes of Kinect V2 Based Measurement System for Shoulder Motions," *Medical Engineering and Physics*, vol. 76, no. 1, pp. 20-31, 2020. https://doi.org/10.1016/j.medengphy.2019.10.017

[13] Cui J., Wang R., Si S., and Hsieh C., "Scaling Up Dataset Distillation to Imagenet-1k with Constant Memory," *in Proceedings of the International Conference on* Machine *Learning*, Honolulu, pp. 6565-6590, 2023. https://dl.acm.org/doi/10.5555/3618408.3618670

[14] Dogan Y., "Which Pooling Method is Better: Max, Avg, or Concat (Max, Avg)," *Communications*

Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, vol. 66, no. 1, pp. 95-117, 2023. https://doi.org/10.33769/aupse.1356138

[15] Erkoc T. and Eskil M., "A Novel Similarity Based Unsupervised Technique for Training Convolutional Filters," *IEEE Access*, vol. 11, pp. 49393-49408, 2023. https://doi.org/10.1109/ACCESS.2023.3277253

[16] Gaspar A., Oliva D., Cuevas E., Zaldivar D., and et al., "Hyperparameter Optimization in a Convolutional Neural Network Using Metaheuristic Algorithms," *Metaheuristics in Machine Learning: Theory and Applications*, vol. 967, pp. 37-59, 2021. https://doi.org/10.1007/978-3-030-70542-8_2

[17] Goodfellow I., Abadie J., Mirza M., Xu B., and et al., "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, vol. 27, pp. 1-9, 2014. https://papers.nips.cc/paper_files/paper/2014/hash/f033ed80deb0234979a61f95710dbe25-Abstract.html

[18] Gronningsaeter Y., Smorvik H., and Granmo O., "An Optimized Toolbox for Advanced Image Processing with Tsetlin Machine Composites," *arXiv Preprint*, vol. arXiv:2406.00704v2, pp. 1-8, 2024. https://arxiv.org/abs/2406.00704v2

[19] Haider U., Hanif M., Rashid A., and Hussain S., "Dictionary-Enabled Efficient Training of ConvNets for Image Classification," *Image and Vision Computing*, vol. 135, pp. 104718, 2023. https://doi.org/10.1016/j.imavis.2023.104718

[20] Hochreiter S. and Schmidhuber J., "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. https://doi.org/10.1162/neco.1997.9.8.1735

[21] Huang G., Zhu Q., and Siew C., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006. https://doi.org/10.1016/j.neucom.2005.12.126

[22] Huang Y., Zhou P., Yang Y., Chen T., and Li N., "Time-Delayed Reservoir Computing Based on a Two-Element Phased Laser Array for Image Identification," *IEEE Photonics Journal*, vol. 13, no. 5, pp. 1-9, 2021. https://doi.org/10.1109/JPHOT.2021.3115598

[23] Hutter F., Lucke J., and Thieme L., "Beyond Manual Tuning of Hyperparameters," *KI-Kunstliche Intelligenz*, vol. 29, pp. 329-337, 2015. https://doi.org/10.1007/s13218-015-0381-0

[24] Jha K., Doshi A., Patel P., and Shah M., "A Comprehensive Review on Automation in Agriculture Using Artificial Intelligence," *Artificial Intelligence in Agriculture*, vol. 2, pp. 1-12, 2019. https://doi.org/10.1016/j.aiia.2019.05.004

[25] Jiao L. and Zhao J., "A Survey on the New Generation of Deep Learning in Image Processing," *IEEE Access*, vol. 7, pp. 172231-172263, 2019. 10.1109/ACCESS.2019.2956508

[26] Kasula B., "Advancements and Applications of Artificial Intelligence: A Comprehensive Review," *International Journal of Statistical Computation and Simulation*, vol. 8, no. 1, pp. 1-7, 2016. https://journals.threws.com/index.php/IJSCS/article/view/214

[27] Kaur J. and Kaur P., "A CNN Transfer Learning-Based Automated Diagnosis of COVID-19 from Lung Computerized Tomography Scan Slices," *New Generation Computing*, vol. 41, no. 4, pp. 795-838, 2023. https://doi.org/10.1007/s00354-023-00232-3

[28] Khan A., Sohail A., Zahoora U., and Qureshi A., "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," *Artificial Intelligence Review*, vol. 53, pp. 5455-5516, 2020. https://doi.org/10.1007/s10462-020-09825-6

[29] Krizhevsky A., *Learning Multiple Layers of Features from Tiny Images*, University of Toronto, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[30] Lecun Y. and Cortes C., The MNIST Database of Handwritten Digits, https://www.lri.fr/~marc/Master2/MNIST_doc.pdf, Last Visited, 2025.

[31] Lecun Y., Bengio Y., and Hinton G., "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015. https://doi.org/10.1038/nature14539

[32] Lecun Y., Bottou L., Bengio Y., and Haffner P., "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998. https://doi.org/10.1109/5.726791

[33] Li J., Lefevre P., and Abdul Majeed A., "Randomness and Interpolation Improve Gradient Descent: A Simple Exploration in CIFAR Datasets," *in Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Guangzhou, pp. 56-59, 2024. https://ieeexplore.ieee.org/document/10771347

[34] Mishra V. and Kane L., "A Survey of Designing Convolutional Neural Network Using Evolutionary Algorithms," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5095-5132, 2023. https://doi.org/10.1007/s10462-022-10303-4

[35] Nichols J., Chan H., and Baker M., "Machine Learning: Applications of Artificial Intelligence to Imaging and Diagnosis," *Biophysical Reviews*, vol. 11, pp. 111-118, 2019. https://doi.org/10.1007/s12551-018-0449-9

[36] Ortiz E., Trigo M., Morillo L., Caparrini F., and Olmos J., "Exploring Deep Echo State Networks

for Image Classification: A Multi-Reservoir Approach," *Neural Computing and Applications*, vol. 36, pp. 11901-11918, 2024. https://doi.org/10.1007/s00521-024-09656-4

[37] Prakash D., Kumar K., and Kumar R., "Hyper-Parameter Optimization Using Metaheuristic Algorithms," *CVR Journal of Science and Technology*, vol. 23, no. 1, pp. 37-43, 2022. https://cvr.ac.in/ojs/index.php/cvracin/article/view/804/648

[38] Rere L., Fanany M., and Arymurthy A., "Metaheuristic Algorithms for Convolution Neural Network," *Computational Intelligence and Neuroscience*, vol. 2016, no. 1, pp. 1-13, 2016. https://doi.org/10.1155/2016/1537325

[39] Rodrigues I., Neto S., Kelner J., Sadok D., and Endo P., "Convolutional Extreme Learning Machines: A Systematic Review," *Informatics*, vol. 8, no. 2, pp. 1-33, 2021. https://doi.org/10.3390/informatics8020033

[40] Srinivasan S., Francis D., Mathivanan S., Rajadurai H., and et al., "A Hybrid Deep CNN Model for Brain Tumor Image Multi-Classification," *BMC Medical Imaging*, vol. 24, no. 1, pp. 1-21, 2024. https://doi.org/10.1186/s12880-024-01195-7

[41] Sun Y., Shi G., Dong W., and Xie X., "MADPL-Net: Multi-Layer Attention Dictionary Pair Learning Network for Image Classification," *Journal of* Visual *Communication and Image Representation*, vol. 90, pp. 103728, 2023. https://doi.org/10.1016/j.jvcir.2022.103728

[42] Talas U., Yuzgec U., and Cubukcu B., "Object Recognizing Robot Application with Deep Learning," *European Journal of Science and Technology*, vol. 2021, no. 31, pp. 127-133, 2021. https://doi.org/10.31590/ejosat.962558

[43] Thadeshwar H., Shah V., Jain M., Chaudhari R., and Badgujar V., "Artificial Intelligence based Self-Driving Car," *in Proceedings of the 4th International Conference on Computer, Communication and Signal Processing*, Chennai, pp. 1-5, 2020. https://ieeexplore.ieee.org/document/9315223

[44] Traore B., Foguem B., and Tangara F., "Deep Convolution Neural Network for Image Recognition," *Ecological Informatics*, vol. 48, pp. 257-268, 2018. https://doi.org/10.1016/j.ecoinf.2018.10.002

[45] Tuba E., Bacanin N., Strumberger I., and Tuba M., "Convolutional Neural Networks Hyperparameters Tuning," *Artificial Intelligence: Theory and Applications*, vol. 973, pp. 65-84, 2021. https://doi.org/10.1007/978-3-030-72711-6_4

[46] Venkataravanappa V., Chowdappa R., Shamanna M., Krishnappa M., and et al., "Conquering Fashion MNIST with CNNs Using Computer Vision by Pretrained Models: VGG19 and RESNET50," *AIP Conference Proceedings*, vol. 3131, no. 1, pp. 1-3, 2024. DOI: 10.1063/5.0229823

[47] Weng Q., Mao Z., Lin J., and Liao X., "Land-Use Scene Classification Based on a CNN using a Constrained Extreme Learning Machine," *International Journal of Remote Sensing*, vol. 39, no. 19, pp. 6281-6299, 2018. https://doi.org/10.1080/01431161.2018.1458346

[48] Wirkuttis N. and Klein H., "Artificial Intelligence in Cybersecurity," *Cyber, Intelligence, and Security*, vol. 1, no. 1, pp. 103-119, 2017. https://www.inss.org.il/wp-content/uploads/2017/03/Artificial-Intelligence-in-Cybersecurity.pdf

[49] Xiao H., Rasul K., and Vollgraf R., "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv Preprint*, vol. arXiv:1708.07747v2, 2017. https://arxiv.org/abs/1708.07747v2

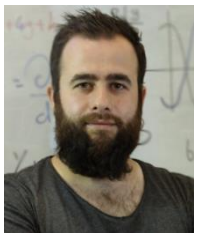
[50] Xin P., Yi T., Yi V., Yu P., and Salam Z., "Convolutional Neural Network for Fashion Images Classification (Fashion-MNIST)," *Journal of Applied Technology and Innovation*, vol. 7, no. 4, pp. 11-16, 2023. https://jati.sites.apiit.edu.my/wp-content/uploads/sites/11/2023/08/Volume7_Issue4_Paper2_2023.pdf

[51] Yang C., Li H., Lin F., Jiang B., and Zhao H., "Constrained R-CNN: A General Image Manipulation Detection Model," *in Proceedings of the IEEE International Conference on Multimedia and Expo*, London, pp. 1-6, 2020. https://doi.org/10.1109/ICME46284.2020.9102825

[52] Yang L. and Shami A., "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," *Neurocomputing*, vol. 415, pp. 295-316, 2020. https://doi.org/10.1016/j.neucom.2020.07.061

[53] Yu T. and Zhu H., "Hyper-Parameter Optimization: A Review of Algorithms and Applications," *arXiv Preprint*, vol. arXiv:2003.05689, pp. 1-56, 2020. https://arxiv.org/abs/2003.05689v1

[54] Zhu W., Miao J., and Qing L., "Constrained Extreme Learning Machine: A Novel Highly Discriminative Random Feedforward Neural Network," *in Proceedings of the International Joint Conference on Neural Networks*, Beijing, pp. 800-807, 2014. https://doi.org/10.1109/IJCNN.2014.6889761



Ugur Talas received his Bachelor's degree from the Department of Computer Education at Duzce University in 2010 and from the Department of Computer Engineering at Bilecik Seyh Edebali University in 2019. He completed his Master's degree in Computer Engineering at Bilecik Seyh Edebali University in 2019. He is currently pursuing a Ph.D. in the Department of Electrical-Electronics and Computer Engineering at Bilecik Seyh Edebali University. Since 2013, he has been working as a Software Developer at the IT Department of Bilecik Seyh Edebali University. His research interests include Deep Learning, Heuristic Algorithms, Image Processing, and Software Technologies.

Burakhan Cubukcu received the Bachelor's degree from the Mathematics Department, Hacettepe University, Ankara, Turkey, in 2012, and Department of Business Administration, Anadolu University, Eskisehir, Turkey in 2010. The Master's degrees from the Information Systems Department, Gazi University, Ankara, Turkey, in 2013. In 2020, he received Ph.D. degree from Department of Electronics and Computer Engineering, Bilecik Seyh Edebali University, Bilecik, Turkey. Since 2021, he has been an Assistant Professor with the Computer Engineering Department, Faculty of Engineering, Bilecik Seyh Edebali University, Turkey. His research interests include Telerehabilitation, Computer Applications, Intelligent Systems, and Deep Learning.

Ugur Yuzgec received the Bachelor's degree from the Electronics and Communication Engineering Department, Yildiz Technical University, Istanbul, Turkiye, in 1995, and the Master's and PhD degrees from the Electronics and Communication Engineering Department, Kocaeli University, Kocaeli, Turkiye, in 1999 and 2005, respectively. From 1998 to 2010, he was a Research Assistant with the Electronics and Communication Engineering Department, Kocaeli University. Since 2020, he has been a Professor with the Computer Engineering Department, Faculty of Engineering, Bilecik Seyh Edebali University, Turkiye. His research interest includes Intelligent Systems and Control, Fuzzy and Neuro-Fuzzy Systems, Meta-Heuristic Algorithms, Unmanned Aerial Vehicles, and Numeric Techniques in Optimization Problems.