Neural Processing Letters (2025) 57:27
https://doi.org/10.1007/s11063-025-11741-1

A Novel Non-iterative Training Method for CNN Classifiers Using Gram–Schmidt Process

Basim Azam[1,2] · Deepthi Kuttichira[1] · Pubudu Sanjeewani[1] · Brijesh Verma[1] · Ashfaqur Rahman[3] · Lipo Wang[4]

Accepted: 19 February 2025 / Published online: 6 March 2025

Abstract
Convolutional neural networks have become prominent machine learning models, particularly in the realm of computer vision, due to their ability to predict and extract robust features from raw image data. CNNs, similar to other neural network models, undergo training via backpropagation, an iterative technique. However, the backpropagation algorithm has notable challenges, including slow convergence, susceptibility to local minima, and hypersensitivity to learning rates. These challenges not only impact the model's accuracy but also make the training process computationally intensive. To address these limitations, We introduce a novel approach that trains the CNN classifier using a non-iterative learning method. The proposed approach involves automatic extraction of pertinent features from the raw-data, followed by the application of Gram–Schmidt process to decompose the feature matrix and determine classifier's weights. The proposed method has shown enhanced predictive accuracy over state-of-the-art models when evaluated on two benchmark datasets, MNIST and CIFAR-10. The extensive experimentation using most cited pre-trained experiments validate the effectiveness of our proposed method.

Keywords Non-iterative · Gram–Schmidt · Convolution neural network

1 Introduction

Machine Learning (ML) models, particularly Convolutional Neural Networks (CNNs), have become predominant in the field of Computer Vision (CV). This dominance is attributed to their ability to automatically learn relevant features directly from raw data, coupled with their exceptional predictive accuracy [1, 2]. Prior to neural network models, features had to be manually engineered from raw data, which was time-consuming and required domain knowledge [3–5]. CNNs have eliminated the need for heavily engineered features. Similar to most other neural networks, CNNs are generally trained using the backpropagation algorithm

✉ Basim Azam
b.azam@griffith.edu.au

1 Griffith University, Brisbane, Australia

2 University of Melbourne, Melbourne, Australia

3 CSIRO, Hobart, Australia

4 Nanyang Technological University, Singapore, Singapore

[6–8]. The backpropagation algorithm is an iterative training method that updates the weights of the network with the goal of minimizing the loss function. Although backpropagation is a convenient method for automatically training the network, it has several drawbacks, such as the risk of getting stuck in local minima, low convergence rate, and high sensitivity to the learning rate, among others [9, 10].

The general structure of a convolutional neural network typically comprises two main components: a feature extraction part and a classification part. The feature extraction part includes multiple layers of convolutional and pooling operations, while the classification part consists of a fully connected feedforward network. In the feature extractor part, the first layers learn low-level features and the subsequent layers learn higher-level features [11]. Thus, the feature extractor part learns the features hierarchically. The Fully Connected (FC) part, which forms the classifier, learns to predict the output based on the extracted features. Both the extracted feature and the classifier part play an important role in the performance of the model [12].

Recent advancements in CNN models have emphasized deepening network architecture by adding more layers, which enables the extraction of more complex, hierarchical features and enhances predictive accuracy. Augmenting the FC section with additional layers has also proven beneficial for accuracy but leads to an exponential increase in the number of parameters, complicating the training process and necessitating larger datasets [13–17]. To mitigate these computational challenges, transfer learning has been employed, utilizing pre-trained models such as VGG16 [18], Resnet50 [19], and LeNet as foundational backbones. These models facilitate feature extraction from raw data, requiring only the retraining of FC layers to adapt to new datasets, thus significantly reducing computational demands. However, despite these efficiencies, many datasets still require extensive retraining of the FC layers, or even complete model retraining, to meet accuracy needs.

Several studies have aimed to enhance accuracy and computational efficiency by substituting the fully connected layers with alternative classifier models. For numerous datasets, superior accuracy was achieved when the FC layers were replaced with classifiers such as Support Vector Machines (SVM) [20] and K-Nearest Neighbours (KNN) [21]. Additionally, efforts to employ non-iterative learning methods, such as the Convolutional Random Vector Functional Link network (CRVFL), Extreme Learning Machines (ELM), and kernelized RVFL networks, have shown potential in boosting efficiency and accuracy without iterative training [20–25]. Despite these advancements, such as ELM's data-driven randomized learning approach [26, 27], which excels in function approximation, they often struggle with generalization issues in classification tasks due to random initialization. This highlights a gap in effectively deploying non-iterative strategies for classification. In Fig. 1, the differences between the traditional CNN architecture and the proposed method integrating Gram–Schmidt orthogonalization are depicted. The traditional CNN architecture, illustrated in the top section, processes raw image data through a series of convolutional layers, culminating in a fully connected layer for classification. This standard approach, while effective, often involves iterative learning which can be computationally intensive and prone to overfitting. Conversely, the proposed method, shown in the bottom section, introduces a novel integration of Gram–Schmidt orthogonalization following the feature extraction layers. This non-iterative learning technique decorrelates the extracted features, thereby reducing redundancy and enhancing the overall classification performance. By eliminating the need for iterative weight updates, the proposed method not only simplifies the training process but also achieves higher accuracy, as demonstrated in our experimental results.

To address these challenges, we introduce a novel approach that leverages CNN's convolutional and pooling layers to efficiently extract features. This method employs a non-iterative

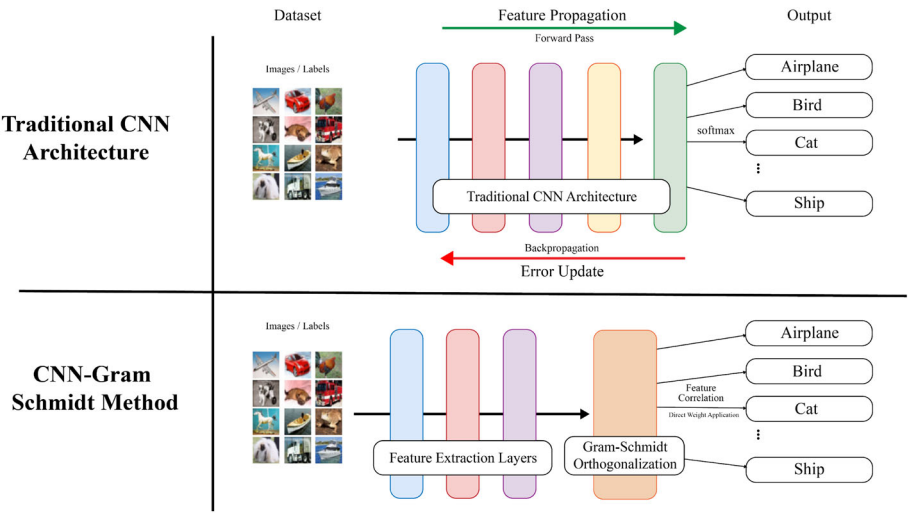

Fig. 1 Comparison of Traditional CNN Architecture with the Proposed Method. The top section shows the standard CNN approach where features are processed through multiple convolutional layers leading to a fully connected layer for classification. The bottom section illustrates the proposed method where features extracted through convolutional layers are orthogonalized using the Gram–Schmidt process before classification, enhancing feature decorrelation and improving classification performance

Gram–Schmidt technique for direct mapping of these features to outputs, as depicted in Fig. 1. Our architecture advances the training process by utilizing Gram–Schmidt Orthogonalization followed by direct weight determination, significantly reducing computational overhead. This paper highlights critical improvements in neural network applications for image recognition, enhancing both efficiency and training efficacy.

1. We propose a novel CNN architecture that automates feature extraction from raw data. Significantly, it leverages the Gram–Schmidt process for a non-iterative approach to learn weights in fully connected layers, facilitating direct mapping from extracted features to the output.
2. We conduct a comprehensive analysis of the proposed model's performance, focusing particularly on accuracy. The detailed evaluation demonstrates the effectiveness of the Gram–Schmidt method in enhancing predictive performance compared to conventional techniques.
3. The validity and robustness of our approach are rigorously tested using two widely recognized baseline benchmark datasets. These evaluations help in establishing a comparative understanding of our model's efficacy.

The remainder of this paper is structured as follows: Sect. 2 elaborates on the proposed method. Section 3 details the datasets employed, describes the experiments conducted, and discusses the results, including a comparative analysis. Finally, Sect. 4 concludes with the key findings from our study and outlines future research directions for the proposed method.

2 Literature Review

Convolutional Neural Networks have been instrumental in advancing the field of machine learning, particularly in computer vision. Since their inception by LeCun et al. in the 1990s, CNNs have transformed image recognition tasks by utilizing a layered architecture that mimics the human visual system. These networks have found widespread application in various domains, including image classification [28–30], object detection [31], and facial recognition [32]. Despite their success, CNNs are traditionally trained using the backpropagation algorithm, an iterative method that, while effective, is accompanied by several significant challenges [3, 33].

Backpropagation is the standard method for training CNNs, involving the calculation of the gradient of the loss function with respect to each weight through the chain rule, followed by iterative weight adjustments to minimize the loss function [34]. Although this approach is widely adopted, it is not without its drawbacks. The iterative nature of backpropagation often leads to slow convergence, particularly in deep networks with many layers, and there is a substantial risk of getting trapped in local minima or saddle points due to the non-convexity of the loss surface [35].

To address these issues, various modifications and enhancements to the backpropagation algorithm have been proposed. Techniques such as momentum [36], adaptive learning rates including AdaGrad, RMSProp, and Adam [37], and regularization methods like dropout [38] have been developed to improve the efficiency and performance of backpropagation. For instance, momentum accelerates convergence by smoothing the updates, while adaptive learning rates dynamically adjust the step size to prevent overshooting and improve stability [6, 39]. Despite these improvements, iterative training methods remain computationally demanding and often require extensive hyperparameter tuning, which can be a significant bottleneck [40].

In contrast to iterative methods, non-iterative training approaches have emerged as a potential solution to the limitations of backpropagation. These methods aim to determine the network's weights in a single pass, eliminating the need for multiple epochs of gradient descent [41]. The Gram–Schmidt process, a mathematical technique for orthogonalizing a set of vectors in an inner product space, has been explored in various linear algebra algorithms [42], and its application in machine learning, particularly for dimensionality reduction and feature selection, has shown promise [43].

The novel training method proposed in this study leverages the Gram–Schmidt process to directly compute the weights of a CNN classifier from the extracted feature matrix, bypassing the need for iterative gradient descent. By ensuring that the feature vectors are orthogonal, this method effectively minimizes redundancy and enhances the discriminative power of the network's features. This structured approach offers a significant advantage over random weight initialization methods, which often suffer from suboptimal generalization performance [44].

Previous non-iterative approaches, such as Extreme Learning Machines [45], have attempted to address the inefficiencies of iterative methods. ELMs, for example, randomly initialize the weights of the hidden layer and then solve a linear system to determine the output weights. While ELMs offer faster training times and are relatively easy to implement, they often fail to generalize well due to the randomness of weight initialization. In contrast, the proposed Gram–Schmidt-based method provides a more deterministic and structured approach to weight determination, leading to more consistent and accurate results.

Iterative methods like backpropagation have dominated CNN training, their limitations in terms of convergence speed, computational efficiency, and sensitivity to hyperparameters

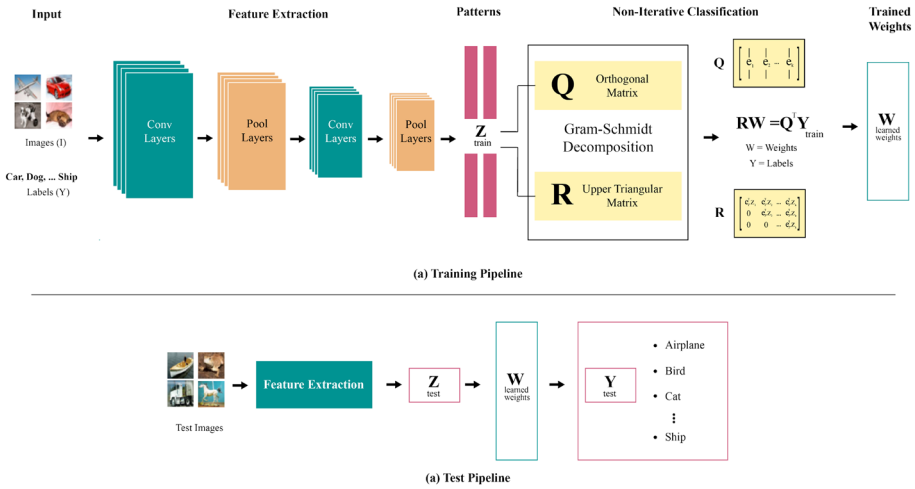

Fig. 2 Illustration of the Proposed CNN-Gram Schmidt Methodology. The upper section demonstrates the feature extraction process through convolutional and pooling layers, followed by the application of Gram–Schmidt orthogonalization on the extracted feature matrix. The lower section highlights the streamlined non-iterative classification using the orthogonalized features for efficient and accurate prediction

have motivated the exploration of non-iterative alternatives. The Gram–Schmidt process, traditionally used in linear algebra for vector orthogonalization, offers a promising foundation for such methods by enabling direct and efficient weight determination. The proposed method's success on benchmark datasets underscores its potential as a viable alternative to backpropagation, particularly in applications where training efficiency and accuracy are critical.

The literature review reveals that while iterative methods such as backpropagation have been the cornerstone of CNN training, they are not without significant drawbacks, including slow convergence, vulnerability to local minima, and sensitivity to hyperparameters. Various improvements and modifications to backpropagation have been proposed, yet the fundamental limitations of iterative approaches remain. Non-iterative methods, including the proposed approach leveraging the Gram–Schmidt process, offer a promising alternative by addressing these challenges through more efficient weight determination.

3 Proposed Methodology

This section outlines a novel image recognition classifier that integrates CNNs with the Gram–Schmidt orthogonalization process, termed CNN-Gram Schmidt (CNN-GS). This hybrid approach is designed to optimize feature extraction and classification efficacy by leveraging the computational efficiency of CNNs for feature extraction and the mathematical precision of the Gram–Schmidt process for orthogonalization. In Fig. 2, we present a detailed overview of the proposed CNN-Gram Schmidt methodology. The figure is divided into two primary sections: the feature extraction phase and the non-iterative classification phase. The *upper section* of the figure illustrates the traditional CNN architecture employed for feature extraction. The input images are processed through a series of convolutional (Conv) and pooling (Pool) layers, which progressively abstract and condense the relevant features. The

resulting feature matrix, denoted as Z_{train}, is then subjected to the Gram–Schmidt orthogonalization process. This process decomposes the matrix into an orthogonal matrix Q and an upper triangular matrix R, effectively decorrelating the features and ensuring that they are linearly independent. The *lower section* of the figure showcases the proposed non-iterative classification method. Here, the orthogonalized features are directly utilized for classification, bypassing the need for iterative weight adjustments. This approach not only simplifies the training process but also enhances the stability and accuracy of the model. By leveraging the Gram–Schmidt orthogonalization, the proposed methodology effectively mitigates the risk of overfitting, particularly when dealing with high-dimensional feature spaces, and ensures robust performance across different datasets.

3.1 Mathematical Framework

The proposed CNN-GS methodology integrates the robust feature extraction capabilities of convolutional neural networks with the mathematical precision of Gram–Schmidt orthogonalization to enhance the classification performance.

3.1.1 Input and Preprocessing

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ represent the dataset, where $x_i \in \mathbb{R}^{d\times h\times c}$ is the i-th image and $y_i \in \{1, \ldots, K\}$ is the corresponding label. Each x_i is standardized to have zero mean and unit variance across the dataset. Mathematically, the preprocessing step can be expressed as:

$$x_i' = \frac{x_i - \mu}{\sigma},$$

where μ and σ are the mean and standard deviation of the pixel intensities across the dataset, respectively.

3.1.2 Feature Extraction via Convolutional Layers

The CNN consists of a series of L layers, each comprising convolutional, activation, and pooling steps. Define a convolutional operation at layer l with kernel $K^{(l)} \in \mathbb{R}^{h\times w\times c_l}$ as:

$$f^{(l)}(x) = \sigma\left(K^{(l)} * x + b^{(l)}\right),$$

where $*$ denotes the convolution operation, $b^{(l)}$ is the bias, and σ is the nonlinear activation function (ReLU). The output of each layer l is then subsampled using a pooling function $P^{(l)}$, typically max pooling, to reduce spatial dimensions:

$$z_i^{(l)} = P^{(l)}\left(f^{(l)}(z_i^{(l-1)})\right),$$

where $z_i^{(0)} = x_i'$.

3.1.3 Orthogonalization via Gram–Schmidt Process

After extracting features $z_i^{(L)}$ at the last convolutional layer, the feature matrix $Z = [z_1^{(L)}, \ldots, z_N^{(L)}]^T$ is formed. The Gram–Schmidt orthogonalization is applied to Z to generate

an orthonormal basis Q:

$$\text{For } j = 1 \text{ to } M:$$
$$v_j = z_j^{(L)} - \sum_{k=1}^{j-1} \text{proj}_{q_k}(z_j^{(L)}),$$
$$q_j = \frac{v_j}{\|v_j\|},$$

where $\text{proj}_{q_k}(v) = \frac{\langle v, q_k \rangle}{\langle q_k, q_k \rangle} q_k$.

3.1.4 Classification and Weight Derivation

With the orthonormal basis Q, the classification weights are derived by minimizing the loss function using the least squares fit:

$$W = (Q^T Q)^{-1} Q^T Y,$$

where Y is the label matrix. This approach reduces to $W = Q^T Y$ when Q is perfectly orthonormal. For a new input x, its class is predicted by:

$$\hat{y} = \text{softmax}(Q W^T f^{(L)}(x)),$$

where $f^{(L)}(x)$ is the feature extraction through the CNN layers, and $W^T f^{(L)}(x)$ projects the features onto the learned weight space.

3.2 Algorithmic Representation

3.2.1 Feature Extraction and Orthogonal Weight Learning

This algorithm details the procedure for training the CNN, extracting features, and applying the Gram–Schmidt process for orthogonalization. The process begins with the initialization of CNN parameters, where features z_i are extracted from each training image x_i using a designated CNN layer l. These features are subsequently compiled into a matrix Z, which is then orthogonalized via the Gram–Schmidt process to generate an orthonormal basis Q and an upper triangular matrix R. The weights W are then calculated using a least squares approach derived from Q and R, ensuring the model learns an effective orthogonal weight matrix that is optimized for classification tasks.

3.2.2 Classification and Testing Using CNN-Gram Schmidt

Following the training, this algorithm leverages the derived feature extractor and weight matrix to classify new images. For each test image, features are extracted and aggregated into a test feature matrix Z_{test}. Utilizing the orthogonal weights W, the algorithm projects the test features onto the label space to predict outputs, and then computes the accuracy against the actual labels to evaluate the model's performance.

Algorithm 1 Feature extraction and orthogonal weight learning

Require: Training dataset $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{k}$
Ensure: Feature extractor f_l, orthogonal weight matrix W
Initialize the CNN parameters.
for each epoch **do**
  for each $(x_i, y_i) \in D_{\text{train}}$ **do**
    $z_i \leftarrow f_l(x_i)$ ▷ Extract features using layer l of the CNN
  end for
end for
$Z \leftarrow [z_1, z_2, \ldots, z_k]^\top$ ▷ Form the feature matrix
$Q, R \leftarrow \text{GramSchmidt}(Z)$ ▷ Orthogonalize features
$W \leftarrow R^{-1}(Q^\top Y)$ ▷ Compute weights using least squares
return f_l, W

Algorithm 2 Classification and testing using CNN-Gram Schmidt

Require: Test dataset $D_{\text{test}} = \{(x_i, y_i)\}_{i=1}^{n-k}$, weights W, feature extractor f_l
Ensure: Predicted labels, Accuracy
$Z_{\text{test}} \leftarrow \emptyset$
for each $x_i \in D_{\text{test}}$ **do**
  $z_{\text{test}} \leftarrow f_l(x_i)$ ▷ Extract test features
  Append z_{test} to Z_{test}
end for
$Y_{\text{test}} \leftarrow W^\top Z_{\text{test}}$ ▷ Predict outputs
Accuracy $\leftarrow$ Evaluate(Y_{test}, y_{test}) ▷ Compare with true labels
return Accuracy, Y_{test}

4 Experiments and Results

4.1 Experimental Setup

We conducted our experiments using a custom Convolutional Neural Network model, implemented using the Keras and TensorFlow frameworks. Our methodology was applied to two benchmark datasets:

- *MNIST* Comprising 60,000 training images and 10,000 testing images, this dataset includes grayscale images of handwritten digits (0–9) with a resolution of 28×28 pixels.
- *CIFAR-10* Consisting of 50,000 training images and 10,000 testing images, this dataset features $32 \times 32 \times 3$ color images across 10 distinct classes.

Our CNN architecture was tailored for each dataset, with variations in the number of neurons in the first layer of the fully connected part to study the impact on feature extraction and classification accuracy. The Gram–Schmidt process was utilized post-feature extraction to derive orthogonal weights for the mapping from features to outputs.

To further assess the generalization of our proposed method, we also evaluated its performance on additional datasets, including the Swedish Leaf Dataset [46] (15 classes), Chest X-ray Dataset [47] (binary classification: normal and pneumonia), and Oxford 102 Flower Dataset [48] (102 classes). These datasets cover diverse classification tasks, spanning botanical classification, medical imaging, and fine-grained object recognition. The results from these datasets serve as additional empirical validation of the effectiveness of our approach in different domains.

Table 1 Model performance on MNIST dataset

	20		500		1000		2000		3000	
Dropped	1		25		50		100		150	
Train Acc (%)	98.70	98.90	99.20	99.40	99.30	99.50	99.40	99.60	99.50	99.70
Test Acc (%)	98.50	98.70	98.90	99.10	99.00	99.20	99.10	99.30	99.20	99.40

Table 2 Model performance on CIFAR-10 dataset

	200		500		1000		2000		3000	
Dropped	10		25		50		100		150	
Train Acc. (%)	85.00	86.00	88.00	89.00	90.00	91.00	92.00	93.00	93.00	94.00
Test Acc. (%)	84.50	85.00	87.00	88.00	89.00	90.00	91.00	92.00	92.00	93.00

4.2 Results and Analysis

4.2.1 Results Overview

The base CNN models for MNIST and CIFAR-10, referred to as "base models," underwent training to extract features from the first layer of the FC part. The number of neurons varied, providing insight into the influence of feature vector size on classification accuracy. The results indicated the elimination of non-activated neurons, enhancing the matrix invertibility necessary for effective QR decomposition.

4.2.2 Detailed Accuracy Tables

For both datasets, we varied the number of original features (neurons in the first layer of the FC part of the base CNN model) from the minimal required for each dataset up to 3,000, reflecting the increased complexity and size of CIFAR-10 compared to MNIST.

4.2.3 Comparative Analysis

The results from Tables 1 and 2 demonstrate that the CNN-Gram Schmidt model typically exhibits improved accuracy compared to the base CNN model, especially at higher feature dimensions. This suggests that the orthogonalization of features may reduce overfitting and improve feature separability, particularly beneficial in handling complex datasets like CIFAR-10.

5 Results and Comparative Analysis

5.1 Initial Training Performance

The Tables 3 and 4 detail the performance of the CNN and CNN-Gram Schmidt models after a single training iteration on the MNIST and CIFAR-10 datasets, showcasing how the feature length impacts model accuracy.

Table 3 Accuracy on MNIST data after training for 1 iteration

Number of features	20	25	30	35	1000
Dropped features	19	22	29	32	990
Train accuracy (%)	98.43	98.42	98.52	98.44	98.74
Test accuracy (%)	98.65	98.79	98.33	98.75	98.87

Table 4 Accuracy on CIFAR-10 data after training for 1 iteration

Number of features	200	400	600	800	1000
Dropped features	135	254	366	540	608
Train accuracy (%)	65.72	66.01	65.80	65.92	66.49
Test accuracy (%)	61.46	59.81	65.05	71.22	66.95

Table 5 Accuracy comparison of proposed method on MNIST

Model	Test accuracy (%)
Proposed method	99.34
ResNet34 + embedded layers [49]	95.33
CBoF + DSH [50]	99.45
F-CNN with Std CNN [9]	99.50
Tsetlin Machine [51]	98.20
Spike-only Feedback [52]	98.10

Table 6 Accuracy comparison of proposed method on CIFAR-10

Model	Test accuracy (%)
Proposed method	90.11
ResNet34 + embedded layers [53]	85.07
CBoF + DSH [50]	88.7–
F-CNN with Std CNN [9]	89.40
Grouped Pointwise Convs. [54]	89.81

A comparative analysis of the proposed CNN-Gram Schmidt method with other state-of-the-art approaches demonstrates its competitive or superior performance. Below are the accuracy comparisons for the MNIST and CIFAR-10 datasets (Table 5).

These findings indicate that the integration of Gram Schmidt orthogonalization into CNN training effectively enhances early training dynamics, leading to more efficient weight adjustments and improved generalization. This innovative approach shows promise in reducing overfitting and in handling high-dimensional data effectively, as evidenced by our results across diverse testing scenarios (Table 6).

5.2 Use of Pre-trained Networks with Gram–Schmidt Process

In addition to the custom CNN architectures developed, we integrated pre-trained neural networks such as VGG16, VGG19, ResNet50, DenseNet121, and DenseNet169 with the Gram–Schmidt orthogonalization process to evaluate the generalizability and scalability of our non-iterative learning approach. These pre-trained networks were chosen due to their

Table 7 Performance comparison of pre-trained networks with proposed Gram–Schmidt method on MNIST dataset in terms of accuracy (%)

No	Method	Pre-trained		Gram–Schmidt	
		Train	Test	Train	Test
1	DenseNet121	99.44	99.36	99.68	99.56
2	DenseNet169	99.44	99.38	99.70	99.56
3	ResNet50	98.62	98.64	99.32	99.21
4	VGG16	99.48	99.37	99.80	99.41
5	VGG19	99.19	99.18	99.75	99.29

Table 8 Performance comparison of pre-trained networks with proposed Gram–Schmidt method on CIFAR-10 dataset in terms of accuracy (%) from intermediate pooling layer

No	Method	Pre-trained		Gram–Schmidt	
		Train	Test	Train	Test
1	DenseNet121	88.72	84.84	89.94	85.76
2	DenseNet169	90.13	85.66	91.21	86.17
3	ResNet50	85.13	80.69	86.90	82.19
4	VGG16	90.35	85.81	91.48	86.54
5	VGG19	87.36	83.93	88.75	85.20

Table 9 Performance comparison of pre-trained networks with proposed Gram–Schmidt method on CIFAR-10 dataset in terms of accuracy (%) from latent pooling layer

No	Method	Pre-trained		Gram–Schmidt	
		Train	Test	Train	Test
1	DenseNet121	88.72	84.84	89.94	85.76
2	DenseNet169	90.13	85.66	91.21	86.17
3	ResNet50	85.13	80.69	86.90	82.19
4	VGG16	90.35	85.81	91.48	86.54
5	VGG19	87.36	83.93	88.75	85.20

established effectiveness in feature extraction, allowing us to test the impact of the Gram–Schmidt method on the extracted feature space.

The convolutional and pooling layers of the pre-trained networks were used to extract features from the CIFAR-10 and MNIST datasets. The fully connected layers were replaced with a Gram–Schmidt orthogonalized layer, which directly determined the classification weights without the need for iterative training.

MNIST Results Table 7 outlines the performance of the pre-trained networks on the MNIST dataset. Unlike the CIFAR-10 results, the accuracies on the MNIST dataset were already high with the pre-trained networks, but the Gram–Schmidt method still provided slight improvements.

DenseNet121 achieved a test accuracy of 99.56% with Gram–Schmidt, compared to 99.36% before. Similarly, ResNet50 improved from 98.64% to 99.21% in test accuracy after applying the Gram–Schmidt method. These results demonstrate that the Gram–Schmidt method can complement the strong feature extraction capabilities of pre-trained networks, yielding competitive results while offering computational advantages by avoiding iterative training.

CIFAR-10 Results Tables 8 and 9 summarize the performance of DenseNet121, DenseNet169, ResNet50, VGG16, and VGG19 when integrated with the Gram–Schmidt method, comparing results from the intermediate and latent layers of the networks.

- Table 8 shows the results from the intermediate pooling layer (Layer 5). The test accuracies of all networks improved slightly when using the Gram–Schmidt method. For instance, DenseNet121's test accuracy increased from 84.84 to 85.76%, and VGG16 saw an improvement from 85.81 to 86.54%.
- Table 9 presents the results from the latent pooling layer (Layer 7). Similar improvements were observed, with ResNet50's test accuracy increasing from 80.69 to 82.19%, and DenseNet169 improving from 85.66 to 86.17%.
- The results suggest that the Gram–Schmidt orthogonalization process enhances the decorrelation of extracted features, improving the robustness of the learned representations on the CIFAR-10 dataset.

The results obtained indicate that the combination of pre-trained networks with Gram–Schmidt orthogonalization yields competitive accuracy rates, rivaling traditional backpropagation methods while offering advantages in terms of computational efficiency. Notably, the performance gains observed on the CIFAR-10 dataset suggest that the method is particularly advantageous for more complex data representations, where feature decorrelation plays a critical role in enhancing classification performance.

6 Analysis

6.1 Theoretical Foundation and Practical Implementation

The integration of the Gram Schmidt orthogonalization process into CNN training represents a significant shift towards non-iterative learning approaches in image recognition. This method is designed to circumvent the limitations commonly associated with backpropagation, particularly in the latter stages of the network training. By decorrelating features prior to weight assignment, the proposed method not only enhances training efficiency but also mitigates the risk of falling into local minima, which are prevalent in traditional training methodologies.

The proposed approach characterizes the process by which the Gram Schmidt method can directly determine weights from decorrelated features, streamlining the computational process and potentially enhancing the model's stability during training.

6.2 Empirical Validation

Empirical evidence from experiments conducted on benchmark datasets, MNIST and CIFAR-10, underscores the effectiveness of the proposed CNN-Gram Schmidt model. The results clearly demonstrate a consistent improvement in classification accuracy over standard CNN models. This substantiates the theoretical claims and underscores the practical utility of the CNN-GS model in real-world applications. To further validate its performance, the method was applied to additional datasets, including the Swedish Leaf Dataset, Chest X-ray Dataset, and Oxford Dataset 102. The results, summarized in Table 10, demonstrate a consistent improvement in classification accuracy after applying the proposed method.

The results section reveals that the CNN-GS model not only achieves higher accuracy compared to traditional CNNs but also illustrates the method's robustness across varied test scenarios.

Table 10 Performance comparison of additional datasets before and after applying the proposed method in terms of accuracy (%)

No	Dataset	Pre-trained		Gram–Schmidt	
		Train	Test	Train	Test
1	Swedish Leaf Dataset [46]	95.20	87.10	97.46	89.60
2	Chest X-ray Dataset [47]	87.62	90.22	96.93	85.26
3	Oxford Dataset 102 [48]	99.85	87.35	99.87	87.45

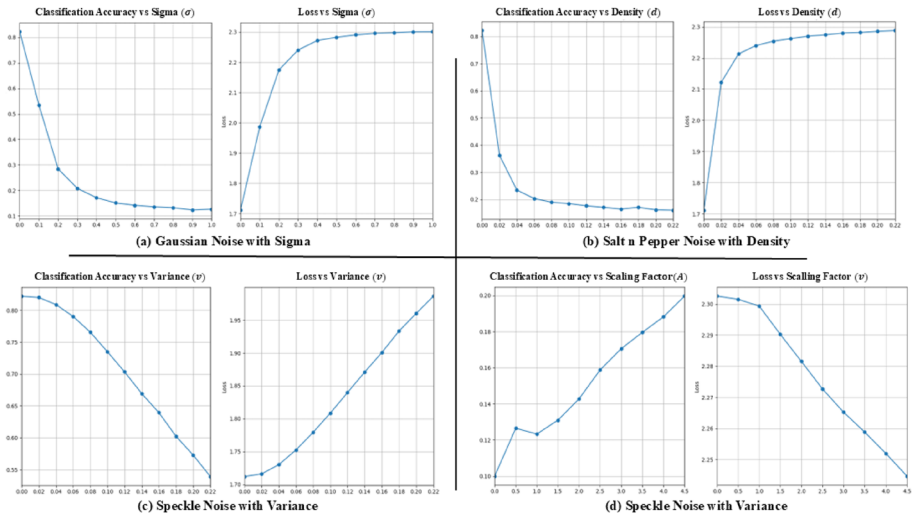

Fig. 3 Robustness analysis on CIFAR-10 using ResNet50 under different noise conditions: Gaussian noise (σ), Salt & Pepper noise (density, d), Speckle noise (variance, v), and Poisson noise (scaling factor, A). Classification accuracy and loss trends are illustrated for each case

6.3 Simplification of Training Procedures

Our proposed method simplifies the training process by partially eliminating the need for iterative weight updates, which is a significant advancement over traditional methods. This simplification makes the CNN-GS model particularly attractive for applications where rapid deployment of models is critical.

The manuscript details how this non-iterative learning approach reduces the computational overhead, facilitating quicker model training without compromising the accuracy.

6.4 Robustness Analysis

To evaluate the robustness of the proposed method, we tested its performance on the CIFAR-10 dataset using the ResNet50 architecture under various noise conditions. These included Gaussian noise, Salt & Pepper noise, Speckle noise, and Poisson noise, with varying parameters. The results, as shown in Fig. 3, provide insights into the sensitivity and resilience of the method:

- Gaussian Noise (σ): Increasing the standard deviation (σ) of Gaussian noise caused a noticeable degradation in classification accuracy while progressively increasing the loss. This indicates a vulnerability of the method to high levels of Gaussian perturbations.
- Salt & Pepper Noise (Density, d): The addition of Salt & Pepper noise with increasing density (d) resulted in a steep decline in accuracy. Concurrently, the loss values rose sharply, showcasing the method's sensitivity to sparse, high-amplitude noise.
- Poisson Noise (Scaling Factor, A): The introduction of Poisson noise with varying scaling factors (A) exhibited an interesting trend. While accuracy slightly improved at lower A values, it declined for higher values.
- Speckle Noise (Variance, v): As the variance (v) of Speckle noise increased, the accuracy steadily declined, while the loss demonstrated a gradual rise. This suggests a consistent performance degradation under this type of multiplicative noise.

These findings highlight the sensitivity of the proposed method to different noise types. Such analysis underscores the importance of robustness in real-world applications where noise is often inevitable. The observed trends further provide valuable insights for potential future work aimed at improving noise resilience in neural networks.

6.5 Enhancement of Model Generalization

One of the standout features of the proposed method is its ability to improve model generalization. By ensuring that features are orthogonal before weight assignment, the CNN-GS model can potentially reduce overfitting-a common challenge in machine learning-thereby enhancing the model's performance on new, unseen data.

Discussions in the manuscript highlight how orthogonalization of features through the Gram Schmidt process minimizes feature redundancy and enhances the generalization capability of the CNN, particularly in complex datasets like CIFAR-10.

6.6 Future Research Directions and Potential Limitations

While the Gram–Schmid process offers a robust mechanism for ensuring feature orthogonality, its computational complexity grows significantly with the dimensionality of the feature space. For extremely large feature vectors, such as those encountered in high-resolution image datasets or deep neural networks with multiple layers, the computational overhead can become a bottleneck. This limitation necessitates further exploration into optimization strategies that balance orthogonality with efficiency.

Future research should focus on developing scalable adaptations of the GS process, such as approximate orthogonalization techniques or hybrid methods that leverage sparsity in feature representations. Additionally, the integration of the CNN-GS model into transfer learning workflows is a promising avenue. By embedding the GS process as a plug-in module within pre-trained architectures like ResNet, DenseNet, and MobileNet, the method's applicability to diverse tasks-including natural language processing, medical imaging, and high-dimensional data analysis-can be evaluated.

Further efforts should also investigate distributed and parallel processing methods to reduce the computational burden in high-dimensional settings. Exploring the trade-off between exact orthogonality and computational efficiency will be critical in determining the practical feasibility of deploying the CNN-GS model in real-world scenarios, particularly in time-sensitive or resource-constrained environments.

The manuscript lays a strong foundation for this exploration, highlighting the model's versatility and potential as a non-iterative learning paradigm for neural networks. With ongoing advancements, the CNN-GS model could redefine efficient training paradigms for next-generation AI applications.

7 Conclusion

In this study, we have introduced a novel methodology that automates feature extraction from raw data and leverages a non-iterative learning technique for efficient classification. Our investigations underscore the critical role of feature vector size in capturing salient information necessary for classification tasks. The efficacy of the feature extraction phase, crucial for the accurate determination of weights in the feature-to-output mapping, becomes particularly vital when implementing non-iterative training strategies, such as the Gram–Schmidt orthogonalization method. Empirical results from our experimentation indicate that the proposed model often exceeds the performance of many current leading models in terms of classification accuracy. This success highlights the potential of combining non-iterative approaches with deep learning frameworks to enhance model efficiency and effectiveness. The research is aimed to examine the influence of features extracted across different convolutional layers on model accuracy. Additionally, we assess the performance of these models against more complex architectures, thereby providing a more comprehensive evaluation of their scalability and robustness.

Acknowledgements This research was supported under Australian Research Council's Discovery Projects funding scheme (Project Number DP210100640).

Author Contributions Basim Azam: conceptualization, coding, implementation, prepared figures, and writing the original draft. Deepthi Kuttichira: conceptualization, review, and editing of the manuscript. Pubudu Sanjeewani: Review and editing of the manuscript. Brijesh Verma: supervision, funding acquisition, and critical review of the manuscript for intellectual content. Ashfaqur Rahman: External investigation and validation. Lipo Wang: External investigation, validation, and critical feedback on experimental results.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare that they have no competing interests.

References

1. Kaur P, Singh SK, Singh I, Kumar S (2021) Exploring convolutional neural network in computer vision-based image classification. In: International conference on smart systems and advanced computing (Syscom-2021)
2. Zhang Y, Zhao D, Sun J, Zou G, Li W (2016) Adaptive convolutional neural network and its application in face recognition. Neural Process Lett 43:389–399
3. Pimsarn N, Dumrongsiri A, Chantrapornchai C (2022) An optimized neural network prediction model for reservoir characterization using shuffled frog-leaping algorithm. Complex Intell Syst 8:5531–5547
4. Kitayama M, Kiya H (2019) Hog feature extraction from encrypted images for privacy-preserving machine learning. In: 2019 IEEE international conference on consumer electronics-Asia (ICCE-Asia), pp 80–82. IEEE
5. David L (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60:91–110
6. Ilesanmi AE, Ilesanmi AE (2021) Methods for image denoising using convolutional neural network: a review. Complex Intell Syst 7:2879–2890
7. Yadav SS, Jadhav SM (2021) Convolutional neural network-based deep learning model for COVID-19 detection using chest X-ray images. Complex Intell Syst 7:3173–3186
8. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8:1–74
9. Sinha T, Verma B (2022) Fast convolutional neural network with iterative and non-iterative learning. Appl Soft Comput 125:109197. https://doi.org/10.1016/j.asoc.2022.109197
10. Zhang S, Yu S, Ding H, Hu J, Cao L (2023) CAM R-CNN: end-to-end object detection with class activation maps. Neural Process Lett 55(8):10483–10499
11. Tiba A, Hajdu A, Giraszi T (2024) Finding efficient graph embeddings and processing them by a CNN-based tool. Neural Process Lett 56(5):226
12. Iqbal S, Ghani MU, Saba T, Rehman A, Nisar K, Damaševičius R (2022) Deep learning model for classifying COVID-19 infection in chest X-ray images. Complex Intell Syst 8:1649–1664
13. Boufssasse A, Hssayni E, Joudar N-E, Ettaouil M (2023) A multi-objective optimization model for redundancy reduction in convolutional neural networks. Neural Process Lett 55(7):9721–9741
14. Giveki D, Soltanshahi MA, Rastegar H (2024) Shape classification using a new shape descriptor and multi-view learning. Displays 79:102636. https://doi.org/10.1016/j.displa.2023.102636
15. Ahamed H, Alam I, Islam MM (2019) Handwritten digit recognition system based on LRM and SVM algorithm. In: International conference on engineering research and education
16. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
17. Paul AN, Yan P, Yang Y, Zhang H, Du S, Wu QJ (2021) Non-iterative online sequential learning strategy for autoencoder and classifier. Neural Comput Appl 33(23):16345–16361
18. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR workshop and conference proceedings, pp 315–323
19. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
20. Niu X-X, Suen CY (2012) A novel hybrid CNN-SVM classifier for recognizing handwritten digits. Pattern Recognit 45(4):1318–1325
21. Wiatowski T, Bölcskei H (2017) A mathematical theory of deep convolutional neural networks for feature extraction. IEEE Trans Inf Theory 64(3):1845–1866
22. Zhang L, Suganthan PN (2016) Visual tracking with convolutional random vector functional link network. IEEE Trans Cybern 47(10):3243–3253
23. Zhang J, Li Y, Xiao W, Zhang Z (2020) Non-iterative and fast deep learning: multilayer extreme learning machines. J Frankl Inst 357(13):8925–8955
24. Henríquez PA, Ruz GA (2018) A non-iterative method for pruning hidden neurons in neural networks with random weights. Appl Soft Comput 70:1109–1121
25. Dalal S, Vishwakarma VP, Sisaudia V, Narwal P (2022) Non-iterative learning machine for identifying covid19 using chest X-ray images. Sci Rep 12(1):11880
26. Zeng Y, Xu X, Fang Y, Zhao K (2015) Traffic sign recognition using extreme learning classifier with deep convolutional features. In: The 2015 international conference on intelligence science and big data engineering (IScIDE 2015), Suzhou, China, vol 9242, pp 272–280
27. Dudek G (2020) Data-driven randomized learning of feedforward neural networks. In: 2020 International joint conference on neural networks (IJCNN), pp 1–8. IEEE

28. Anuse A, Vyas V (2016) A novel training algorithm for convolutional neural network. Complex Intell Syst 2:221–234
29. Jiang Y, Xu Z, Jiang Y, Zheng Y, Zhu Y (2024) A novel Voronoi-based convolutional neural network framework for image classification. Complex Intell Syst 10:1–14
30. Bayoudh K, Knani R, Hamdaoui F, al (2023) A review of machine learning and deep learning for object recognition and image classification. MDPI
31. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
32. Li Y, Wang S, Zhao Y, Ji Q (2013) Simultaneous facial feature tracking and facial expression recognition. IEEE Trans Image Process 22(7):2559–2573
33. Hecht-Nielsen R (1992) Theory of the backpropagation neural network. Wiley
34. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. In: Nature, vol 323, pp 533–536. Nature Publishing Group
35. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
36. Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. USSR Comput Math Math Phys 4(5):1–17
37. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
38. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
39. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning, pp 1139–1147. PMLR
40. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. In: Neural networks: tricks of the trade, pp 437–478. Springer
41. Huang G, Li Y, Pleiss G, Liu Z, Hopcroft J, Weinberger KQ (2017) Snapshot ensembles: train 1, get m for free. arXiv preprint arXiv:1704.00109
42. Golub GH, Van Loan CF (1996) Matrix computations. Johns Hopkins University Press
43. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
44. Schmidt WF, Kraaijveld MA, Duin RP (1992) Feedforward neural networks with random weights. In: International conference on pattern recognition, pp 1–5. IEEE
45. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
46. Söderkvist I (2001) Swedish leaf dataset. https://www.cvl.isy.liu.se/research/datasets/swedish-leaf/
47. Kermany DS, Zhang K, Goldbaum M (2018) Labeled optical coherence tomography (OCT) and chest X-ray images for classification. https://data.mendeley.com/datasets/rscbjbr9sj/2
48. Nilsback M-E, Zisserman A (2008) Automated flower classification over a large number of classes. In: Proceedings of the Indian conference on computer vision, graphics and image processing. https://www.robots.ox.ac.ukvgg/data/flowers/102/
49. Chenhui Y, Chunling C (2018) A transfer learning method based on residual block. In: 2018 IEEE 9th international conference on software engineering and service science (ICSESS), pp 807–810. https://doi.org/10.1109/ICSESS.2018.8663793
50. Passalis N, Tefas A (2019) Training lightweight deep convolutional neural networks using bag-of-features pooling. IEEE Trans Neural Netw Learn Syst 30(6):1705–1715
51. Abeyrathna KD, Bhattarai B, Goodwin M, Gorji SR, Granmo O-C, Jiao L, Saha R, Yadav RK (2021) Massively parallel and asynchronous Tsetlin machine architecture supporting almost constant-time scaling. In: Proceedings of the 38th international conference on machine learning. Proceedings of machine learning research, vol 139, pp 10–20. PMLR
52. Park J, Lee J, Jeon D (2019) 7.6 A 65 nm 236.5 nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback. In: 2019 IEEE international solid-state circuits conference-(ISSCC), pp 140–142. IEEE
53. Chenhui Y, Chunling C (2018) A transfer learning method based on residual block. In: 2018 IEEE 9th international conference on software engineering and service science (ICSESS), Beijing, China, pp 807–810. IEEE
54. Schwarz Schuler JP, Romani S, Abdel-Nasser M, Rashwan H, Puig D (2022) Grouped pointwise convolutions reduce parameters in convolutional neural networks. MENDEL 28:23–31