Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 2 (2025)

Optimized Convolutional Neural Network Architecture For Image Classification

A. Jayalakshmi[1], S.R. Tharan[2], P. Roshan[3], M. Dharani[4]

[1] Assistant Professor

[2,3,4] Student

[1,2,3,4] Department of Computer Science and Engineering,

Dr. Mahalingam College of Engineering and Technology, Pollachi.

Abstract:- An optimized Convolutional Neural Network (CNN) is employed for the image classification task using the CIFAR-10 dataset as the foundational dataset. The initial layer of the CNN preprocesses the dataset to the training data. To categorize the images in the dataset into ten classes under the labels aircraft, bird, cat, deer, dog, frog, horse, ship, truck and vehicle. The CNN model architecture is made up of convolutional layers, max-pooling layers and fully connected layers. The activation functions used are ReLU and sigmoid. The proposed system has been trained over 10 epochs with the usage of binary cross-entropy loss and the Adaptive Moment Estimation (Adam) as the optimizer. When tested, the model's test accuracy is 94.32 percent. This paper evaluates the binary classification between images classifying between real and AI generated shows how CNN performs image classification tasks efficiently.

Keywords*: Accuracy, CIFAR-10 Dataset, Convolutional Neural Network, Image Classification.*

1. Introduction

Computer vision has seen a revolution in recent years using the deep learning models and Convolutional Neural Networks (CNNs). Image classification, which involves categorizing images into predefined labels, has emerged as one of the most challenging and widely studied problems in artificial intelligence. CNNs are perfect for applications like image recognition, object detection and scene segmentation because of the ability to learn hierarchical features automatically from raw images. The CIFAR-10 dataset, which includes 60,000 colour 32x32 photos in ten classes, is one of the most frequently used benchmarks for training and testing image classification algorithms. This work involves investigating the classification of image data from the CIFAR-10 dataset using optimised CNN. The primary goal of this paper is to show how CNNs can be used for image classification tasks and assess how they perform when applied over real-world datasets.

2. Objectives

The objective of this paper is to develop an optimized convolutional neural network architecture for image classification tasks. As artificial intelligence continues to evolve around the field of synthetic image generation by using latent diffusion models and other image generation models. Hence, there exist many synthetically generated images all over the Internet. The real images correspond to the real or natural images on the internet, while the fake images represent the synthetically generated images by using a latent diffusion model. People find it difficult to differentiate between real and synthetic images.

Convolutional neural networks are derived from artificial neural networks. Image classifications can be performed more efficiently by using a convolutional neural network to detect various features in images. In this paper we have designed an optimized convolutional neural network architecture that helps to classify the images as real or fake. The key objective of this paper is to make the convolutional neural network architecture differentiate between real and fake images efficiently.

3. Methods

The methods section outlines the various steps undertaken in the experiment, covering dataset overview and key stages such as data preprocessing, model architecture design and evaluation. The experiment's processes are described in detail in the methodology section, which also covers important topics including data preprocessing, model architecture design and evaluation. This section is divided into five main parts: Overview of CIFAR-10 dataset, overview of the CIFAKE dataset, data collection, synthetic data generation and image classification. Beginning with 3.1 and 3.2, it provides an overview of the datasets used, 3.3 shows the data collection process, which involves loading and organizing the dataset into a structured directory format. To ensure consistent input for the model, the photos are rescaled to a range of [0, 1] to standardize pixel values. Section 3.4 focuses on synthetic data generation and its use case. Section 3.5 discusses the image classification process, where a CNN architecture automatically learns the spatial feature hierarchies in the input images. An outline of the CNN layers utilized in this paper is given in this section. These layers are intended to identify significant patterns and carry out efficient image classification using the trained dataset.

3.1 Overview of CIFAR-10 Dataset

For training machine learning models, especially for image classification problems, the CIFAR-10 dataset is the most popular. Ten classes of sixty thousand colour images, each measuring 32x32 pixels, are included. Two subsets of the dataset are created that each include 10,000 of testing images and 50,000 training images. A fair distribution of categories, including Airplane, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck and Vehicle are achieved by each class comprising of 6,000 images. The CIFAR-10 dataset is perfect for deep learning models with a wide range of images of different object sizes, positions and background complexities.

3.2 Overview of CIFAKE Dataset

The CIFAKE dataset was created using the synthetic images generated using CompVis SD, an open-source LDM. This dataset comprises of AI generated synthetic images and real images from CIFAR- 10 dataset each having 60,000 images. This dataset has images structured and labeled equivalent to the CIFAR-10 dataset’s classes.

3.3 Data Collection

The process of data collection begins by loading the CIFAKE dataset from Kaggle and organizing it into a structured directory format. The images in the dataset are separated and images are loaded using Keras’s ImageDataGenerator for both preprocessing and normalisation.

The following steps are followed to properly load and prepare the data:

Step 1, Directory Setup: The images are organized into subfolders corresponding to their respective labels. Each subfolder represents one of the 10 classes in the dataset, ensuring an organized structure for efficient data loading.

Step 2, Preprocessing: Pixel values are divided by 255 to rescale the images to the range [0, 1]. By standardizing the input data, this normalization makes it possible for the model to learn more effectively.

3.4 Synthetic Data Generation

Synthetic data generated through prompts into CompVis SD a Latent Diffusion Model by huggingface. In this paper, the public release of the CIFAKE dataset has been employed for the model training and testing. These synthetic data are a vital resource for ensuring that the CNN model identifies the features of AI-generated images.

3.5 Image Classification

In this paper, the CNN architecture is employed for image classification. Learning the spatial hierarchies of characteristics from input images is a good fit for a CNN. The proposed architecture is made up of various kinds of layers. The layers in the convolutional neural network architecture created for the image classification job are thoroughly explained in this section. Convolutional, max-pooling and fully connected layers are the layers that make up the architecture. Each of these layers is essential to extract features from the images and eventually make predictions. The CIFAKE dataset is made up of small colour images of 32x32 pixels. The architecture is specifically made to process these images effectively and categorize them into one of the 10 categories. A total of three convolutional layers with 160 convolution filters are used in this proposed architecture.

3.5.1 Convolutional Layers Architecture

The basic units of a CNN are called convolutional layers. These layers enable the network to learn spatial hierarchies of features by applying filters, also called kernels, to the input image using the sliding window method. Low-level features like edges, textures and basic forms can be detected by these filters. Important visual patterns from the input data are highlighted by the feature map produced by the convolutional layer as they process the images. ReLU as the activation function comes after each convolutional layer. Learning complicated patterns requires non-linearity, which is introduced into the network by these ReLU activation functions. The network would essentially act like a linear model in the absence of non-linearity, which would restrict its ability to discover intricate links in the data. The network learns and propagates gradients more effectively during training, thanks to the rectified linear unit, which converts any negative values to zero while maintaining positive values. In feature extraction, the convolutional layers are essential because they let the model gradually create increasingly abstract representations of the input image. The features identified by the convolutional layers get increasingly intricate and sophisticated as the network gets deeper.

The CNN acts as the first convolutional network in the entire system. It may be operationally generalized as follows:

For an image with a filter matrix W and dimension x:

$$(x * W)(i,j) = \sum_{m=1}^{M}\sum_{n=1}^{N} x(i+m-1, j+n-1)w(m,n)$$

where (m, n) is the location of the filter, W and (i, j) is the output of the feature map. Then the output is obtained by applying an activation function f,

Rectified Linear Unit, f(x) = max(0, x).

3.5.2 Max-Pooling Layers

In CNN architectures, max-pooling layers are included after each convolutional layer. These layers are intended to preserve the most crucial information while decreasing the spatial dimension of the feature map generated by the convolutional layer. The key purpose of max-pooling layer is downsampling, which lowers the feature maps' resolution. To do this, the feature map is divided into smaller portions and a representative feature for each region is chosen by taking the maximum value from each region. Max-pooling helps avoid overfitting and lowers the computational cost by reducing the number of parameters that must be processed. Max-pooling creates a type of translation invariance by shrinking the feature maps, which makes the network less susceptible to little distortions or translations of the input image. As a result, the model is stronger and equipped to generalize data.

3.5.3 Fully Connected Layers

Fully connected layers are placed after convolutional layers and max-pooling layers, which use the features taken from the input image to generate predictions. These layers' job is to classify the images into one of the predetermined categories using high-level attributes that the previous layers have learned. As every unit in the fully connected layer is connected to every other unit of other layers, the model can use the learned features to make intricate judgments. The output layer, the last fully connected layer, generates probabilities for ten classes in the CIFAKE dataset using a Sigmoid activation function. The Sigmoid function exponentiates each score and normalizes them so that they add up to one to transform the raw output score into generalised probabilities. The image's anticipated label is chosen from the class with the highest likelihood. Sigmoid makes the model acceptable for multi-class classification applications by guaranteeing that it produces probability distribution across classes.

In summary, the network architecture is designed to progressively learn spatial features through convolutional layers that reduce spatial dimensions while retaining important features via max-pooling layers and finally, make a classification decision using fully connected layers. This makes the model effectively learn from the training dataset and correctly categorize the images.

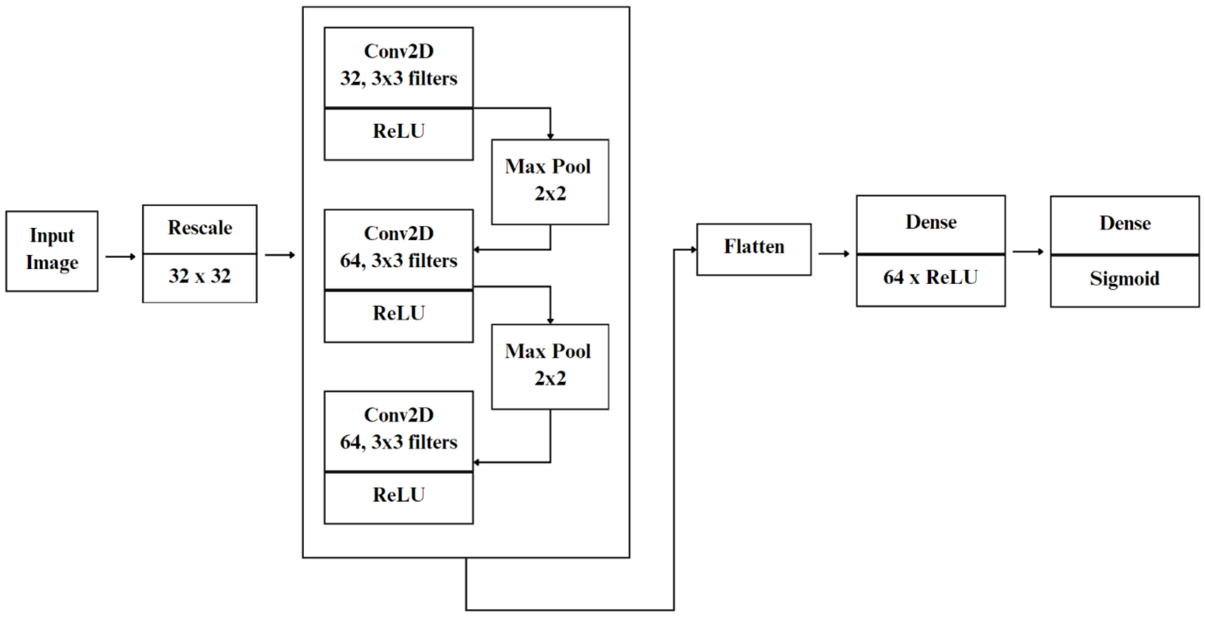

FIGURE 1. Optimised Convolutional Neural Network Architecture

4. Results

Accuracy and loss evaluation charts are used to examine CNN model performance on the CIFAKE dataset. While the model accuracy graph shows an increasing trend, with a consistent improvement in train and test accuracy across epochs, reaching roughly 94.32%. While the model loss shows a gradual decline in subsequent epochs during training and the test loss shows a steady decline on the graph which shows a positive sign of better model performance. Overall, the findings support that the optimised CNN architecture performs great for image classification tasks.

4.1 Evaluation Metrics

In this study, the architecture's objective is to determine if image is real or one produced by the Latent Diffusion Model (LDM). This is a binary classification problem. A value nearer either of the two values indicates the possibility of that class, as 0 for the "FAKE" class and 1 for the "REAL" class. While the network's objective is to minimize the loss of binary cross-entropy through backpropagation. Using a wide range of classification metrics to validate the model performance. This includes Precision that enables the examination of false-positives and quantifies the proportion of predictive positive cases that are positive:

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

Analysis of false-negative predictions is made possible by the recall, which is a measurement of the number of correctly predicted positive cases:

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

As with AI detection, this test is especially crucial. In this instance a false negative case would showcase the usage of artificial intelligence to create synthetic image.

Lastly, the F-1 score is taken into account,

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

This is a unified metric of recall and precision.

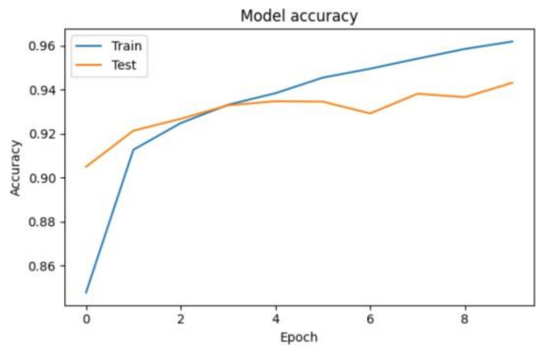

FIGURE 2. Model Accuracy graph visualizes Epoch vs. Accuracy for the computed Model after 10 epochs

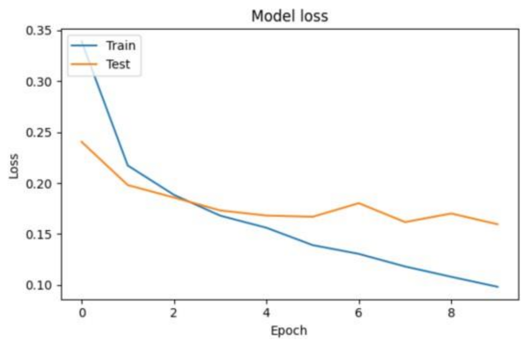

FIGURE 3. Model Loss Graph visualizes Epoch vs. Loss for the computed Model after 10 epochs

Evaluation Metrics	**Optimized CNN Architecture**
Test Accuracy	0.9432
Precision	0.50
Recall	0.50
F1-Score	0.50

TABLE 1. Proposed CNN Architecture Validation across Multiple Evaluation Metrics

The Optimised CNN Architecture as proposed in this paper is evaluated under various evaluation metrics such as the Test Accuracy, Precision, Recall and F1-Score with values is shown in Table 1. This table is used to examine the model performance and compute its efficiency. These findings indicate that the model perform well on classifying the images as real or synthetically generated. Since, the model trained on a consistent batch size of 64 and having the image sizes 32x32 pixel, provides consistency throughout model training and testing across the dataset. Thereby maintaining consistency, the model proving to provide a test accuracy of 94.32 percent.

5. Discussion

This work demonstrates CNNs' potential for binary image classification tasks, namely in differentiating between artificial intelligence-generated synthetic images and genuine images. The use of the optimally required convolution filters, under an optimised convolutional layer architecture produced a competitive accuracy and insightful model predictions.

In order to improve classification performance even further, future research will investigate the use of sophisticated systems like Vision Transformers and attention techniques. A more thorough assessment of the suggested method will also be possible by enlarging the dataset to include more varied classes and higher-resolution photos. To increase the work's relevance to other domains, the incorporation of domain-specific datasets—like satellite or medical imagery will also be examined.

References

[1] J. J. Bird, "CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images," 2024.

[2] J. Bharadiya, "Convolutional Neural Networks for Image Classification," Int. J. Innov. Res. Sci. Eng. Technol., 2023.

[3] H. Mane et al., "Image Classification using Deep Learning," Int. J. Eng. Technol., 2018.

[4] K. Roose, "An AI-generated picture won an art prize. Artists aren't happy," New York Times, vol. 2, pp. 2022, Sep. 2022.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esserand B. Ommer, "High-resolution image synthesis with latent diffusion models," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022.

[6] G. Pennycook and D. G. R and "The psychology of fake news," Trends Cogn. Sci., vol. 25, no. 5, pp. 388-402, May 2021.

[7] D. Deb, J. Zhangand A. K. Jain, "AdvFaces: Adversarial face synthesis," in Proc. IEEE Int. Joint Conf. Biometrics (IJCB), Sep. 2020.

[8] Y. Zhang, et al., "Efficient Channel Attention Networks," IEEE Trans. Image Process., vol. 30, pp. 6362-6374, 2021.

[9] Z. Liu, et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2021.

[10] M. Tan and Q. V. Le, "EfficientNetV2: Smaller and Faster Convolutional Networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2021.

[11] S. Woo, et al., "CBAM: Convolutional Block Attention Module," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020.

[12] A. Dosovitskiy, et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[13] Y. Touir, et al., "Optimizing Image Classification: Automated Deep Learning Architecture Crafting with Network and Learning Hyperparameter Tuning," PMC, 2023.

[14] A. Guernine, et al., "Optimized Training for Convolutional Neural Network Using Enhanced Grey Wolf Optimization Algorithm," Informatica, vol. 32, no. 1, pp. 39-52, 2021.

[15] H. Wu, et al., "Visual Transformers: Token-based Image Representation and Processing," Front. Inf. Technol. Electron. Eng., vol. 22, no. 7, pp. 1104-1116, 2021.